Bridging the Gap: Verlustfreie und sichere Umwandlung von XML-Datenstrukturen ins JSON-Format

Jan Holthuis
Lehrstuhl für Netz-und Datensicherheit
Ruhr-Universität Bochum
Matrikelnummer 108 009 215 809
jan.holthuis@ruhr-uni-bochum.de

***Zusammenfassung*—Bei XML und JSON handelt es sich um – insbesondere im Mobile- und Web-Bereich konkurrierende – menschenlesbare Formate für die Speicherung und den Austausch hierarchisch strukturierter Daten. Je nach eingesetzter API, Programmiersprache oder Programmbibliothek kann es dabei sinnvoll sein, die Daten zunächst in das jeweils andere Format zu überführen. In der geplanten Bachelorarbeit soll sich mit der verlustfreien Translation von XML-Daten in JSON-Strukuren befasst werden. Dazu sollen bestehende Verfahren auf Genauigkeit und Sicherheit hin untersucht werden. Schlussendlich soll in der Arbeit ein verlustfreies Verfahren zur Übersetzung von XML-Daten in JSON sowie in Rückrichtung entwickelt und vorgestellt werden.**

Schlagworte*—*XML, JSON, Konversion, Konvertierung, Abbildung, Datenstrukturen, Sicherheit.

I. Einleitung

Für den implementationsunabhängigen Austausch von zugleich menschen- als auch maschinenlesbaren Daten hat sich die *Extensible Markup Language (XML)* bewährt. In bestimmten Bereichen wie Web-APIs hat die *JavaScript Object Notation (JSON)* das bewährte XML-Format inzwischen jedoch überflügelt.

Dabei kann neben der spezifischen Situation auch die eingesetzte Programmiersprache, die Unterstützung durch das zugrundeliegende Framework oder die persönliche Präferenz des Entwicklers den Ausschlag geben, welches Format ein Webservice oder eine Programmbibliothek unterstützt.

Zwecks Interoperabilität zwischen verschiedenen Teilen einer Anwendung kann es daher notwendig werden, Daten von einem Format temporär in das jeweils andere zu überführen und später wieder in das ursprüngliche Format zu bringen.

Dabei sollen Daten, die von der Anwendungslogik nicht verändert wurden, auch nach der Rückübersetzung ins Ursprungsformat unverändert bleiben. Damit dies gewährleistet ist, darf das zugrunde liegende Konversionverfahren keine Informationen bei der Umwandlung verwerfen – es muss also verlustlos arbeiten.

Das konkrete Konversionsverfahren ist dabei abhängig vom jeweiligen Ausgangsformat, d. h. die Umwandlungsverfahren für die beiden Richtungen

1) JSON → XML → JSON und
2) XML → JSON → XML

haben jeweils eigene Anforderungen und sind getrennt voneinander zu betrachten.

Während Abbildung von beliebigen JSON-Datenstrukturen in XML zumindest bei oberflächlicher Betrachtung trivial erscheint, ist dies beim verlustlosen Transfer von XML-Daten ins JSON-Format keineswegs der Fall.

Daher soll in der geplanten Bachelorarbeit verschiedene Verfahren analysiert werden, die beliebige XML-Daten in JSON abbilden und aus der resultieren JSON-Datenstruktur wieder XML-Dokument erstellen können. Sollte keines der analysierten Verfahren den vorher aufgestellten Kriterien für eine zuverlässige und sichere Umwandlung genügen, wird ein eigener Abbildungsalgorithmus entwickelt, bei dem dies der Fall ist.

II. Motivation

Das Aufkommen des sogenannten *Web 2.0* und die zunehmenden Vernetzung durch das *Internet of Things (IoT)* ging mit einer erhöhten Verfügbarkeit von öffentlichen Web-APIs einher. Als Datenformat wird dabei häufig JSON oder XML verwendet.

Neben XML-basierten Webservices, die beispielsweise das SAML-Framework, SOAP oder XML-RPC verwenden, wird XML auch in einer Vielzahl weiterer Einsatzbereiche eingesetzt. So dient XML den Dateiformaten RSS/ASF, MathML, Scalable Vector Graphics (SVG) oder XHTML als Basis. Auch die gängigen Office-Dateiformate – das *Open Document Format*, Microsofts *Office Open XML* und Apples *iWorks* – bauen auf XML auf.

Inzwischen gewinnt jedoch *JSON* vor allem im Mobile- und Web-Bereich immer mehr an Bedeutung. Laut dem API-Verzeichnis *ProgrammableWeb* unterstützten im Jahr 2013 ca. 60% aller neu hinzugefügten APIs das JSON-Format, während XML im selben Zeitraum lediglich von 37% der neuen APIs unterstützt wurde.[4]

JSON ist bei bestimmten Aufgaben in puncto Geschwindigkeit und Ressourcenauslastung deutlich effizienter[10] als XML. Inzwischen setzen auch einige populäre NoSQL-Datenbanken wie *CouchDB* oder *MongoDB* auf *JSON* zur

Speicherung. Auch MySQL verfügt seit Version 5.7.8 über einen nativen JSON-Datentyp.

Die Umwandlung zwischen den beiden Formaten kann aus vielen Gründen notwendig werden. Soll beispielsweise ein SOAP-Webservice als moderne JSON-REST-Ressource angeboten werden, muss zwischen XML und JSON konvertiert werden. Auch bei der Speicherung von XML-Daten in den o.g. NoSQL-Datenbanken ist dies der Fall. Zudem ist die Unterstützung der Formate durch Programmiersprachen, Frameworks und Applikationen nicht immer gleich gut.

III. Vorgehensweise

Im Verlauf der Arbeit sollen verschiedene generische XML-zu-JSON-Konversionsverfahren miteinander verglichen werden. Anbieten würden sich dazu evtl.

- die *BadgerFish*-Konvention[1],
- die *Parker*-Konvention[2],
- die *Google Data (GData)*-Konvention[3],
- die *JSON Markup Language (JsonML)*[4],
- die *x2js*-Bibliothek[5] (JavaScript, Apache 2.0-Lizenz),
- die *Apache camel-xmljson*-Bibliothek[6] (Java, Apache 2.0-Lizenz),
- die *jxon*-Bibliothek[7] (JavaScript, GNU Public License 3.0) und
- die *pesterfish*-Bibliothek[8] (Python, MIT-Lizenz).

Auch der *IBM DataPower*-Gateway unterstützt XML/JSON-Umwandlungen, jedoch scheint nur die Richtung

$$\text{JSON} \rightarrow \text{JSONx (XML)}^{9} \rightarrow \text{JSON}$$

möglich zu sein. Falls die Umwandlung auch mit dem Ausgangsformat XML möglich ist, wäre das proprietäre Übersetzungsverfahren des *IBM DataPower*-Gateways ebenfalls Gegenstand der Untersuchung.

A. Durchführung

Um eine faire und objektive Bewertung aller Konversionsverfahren zu ermöglichen, werden vorab bestimmte Kriterien aufgestellt, anhand derer die Verfahren bewertet werden sollen. Geprüft werden könnte beispielsweise, ob

- alle getesteten Dokumente umgewandelt werden, ohne das der Konverter abstürzt,
- die aus der Konversion resultierenden Daten wohlgeformtes oder valides JSON bzw. XML sind,
- die Umwandlung verlustfrei abläuft,
- die Konversion ohne zusätzliche Angabe von Metadaten über das reine XML- bzw. JSON-Dokument hinaus funktioniert und
- die Konversion nicht anfällig für die bekannten Angriffe auf XML-Parser ist.

Im Anschluss an das Aufstellen der Kriterien werden XML-Testdokumente erstellt, anhand derer deren Konversionsqualität bzw. Verlustlosigkeit überprüft werden sollen. Jedes XML-Dokument soll dabei die Abbildung bestimmter XML-Features in JSON testen (Attribute, Text-Node-Reihenfolge, etc.).

Indem die Test-XML-Dokumente zunächst vor und nach einem XML-Round-Trip (Umwandlung von XML in JSON, danach Rückumwandlung des resultierenden JSON in XML) verglichen werden, lässt sich feststellen, ob das jeweilige XML-Merkmal fehlerfrei abgebildet wird. Um variierende Darstellungen logisch äquivalenter XML-Dokumente Rechnung zu tragen, werden die zu vergleichenden XML-Dokumente vor dem Vergleich mittels *XML Canonicalization (C14N)*[1] in eine einheitliche Form gebracht.

Auch die Sicherheit der Konversionsverfahren soll getestet werden. Dazu werden Testdokumente erstellt, die die Verwundbarkeit gegenüber allen bekannten XML-Entity-Angriffen[9, 11] wie z.B.

- *Exponential Entity Expansion* (“Billion Laughs”),
- *Quadratic Blowup Entity Expansion*,
- *External Entity Expansion* (“XXE”, sowohl für lokale Pfade als auch URLs)[12] und
- *DTD Retrieval*

überprüfen.

Zuletzt soll mithilfe eines umfangreichen und komplexen XML-Dokuments (ggf. auch mehrerer Test-Dokumente) die Performance der Umwandlung bestimmt werden. Dazu wird das selbe Test-Dokument mehrmals umgewandelt und im Anschluss das arithmetische Mittel der Zeitdauer für die Konvertierung von XML zu JSON bzw. JSON zu XML ermittelt.

Indem die Umwandlungsergebnisse mit einem Prüfprogramm (einem sog. “Linter”) analysiert werden, wird sichergestellt, das es sich dabei um wohlgeformte oder valide XML- bzw. JSON-Dokumente handelt. Als Programme kämen dabei z. B. Tools wie *demjson/jsonlint*[10], *xmllint*[11] oder *XMLStarlet*[12] in Betracht.

B. Erarbeitung eines Umwandlungsalgorithmus

Sollte sich bei der Durchführung der Tests herausstellen, das keines der getesteten XML-zu-JSON-Konversionsverfahren den zuvor aufgestellten Kriterien genügt, soll ein eigener Abbildungsalgorithmus für XML-Datenstrukturen im JSON-Format erarbeitet werden

[1] http://www.sklar.com/badgerfish/
[2] https://github.com/open311/xml-tools/tree/master/Parker/xml2json-xslt
[3] https://developers.google.com/gdata/docs/json
[4] http://www.jsonml.org/
[5] https://github.com/abdmob/x2js
[6] http://camel.apache.org/xmljson.html
[7] https://github.com/tyrasd/jxon
[8] https://bitbucket.org/smulloni/pesterfish/overview
[10] http://deron.meranda.us/python/demjson/
[11] http://xmlsoft.org/xmllint.html
[12] http://xmlstar.sourceforge.net/

Dabei scheint es sinnvoll, das vielversprechendste der zuvor geprüften Verfahren als Basis zu nutzen und die fehlenden Features und Modifikationen zu ergänzen.

Falls es im gesetzten zeitlichen Rahmen einer Bachelorarbeit möglich ist, eine Referenzimplementierung des Algorithmus zu erstellen, soll diese wie alle anderen Verfahren evaluiert werden.

C. Vorläufige Gliederung

Nach der Einleitung wird zunächst eine Einführung in die Struktur der Formate XML und JSON gegeben (wobei die Formate allerdings selbstverständlich nicht erschöpfend dargestellt werden können, da dies den Rahmen sprengen würde). Im Zuge dessen werden ebenfalls die wichtigsten Unterschiede zwischen beiden Formaten aufgezeigt (z. B. JSON-Arrays, XML-Attribute, Duck Typing, etc.).

Daran anschließend wird zunächst das Ziel konkret definiert, d. h. es werden überprüfbaren Kriterien aufgestellt, denen ein Konversionsverfahren genügen muss. Dazu gehört beispielsweise, Verlustlosigkeit genauer zu definieren (vor allem im Bezug auf Fragen wie "Sind Kommentare relevant?"). Hier wird auch ein Abschnitt zu Angriffen (*XXE*, *Billion Laughs*, etc.) eingefügt, gegen den der jeweilige Konverter gewappnet sein soll.

Dann werden die verschiedenen bestehenden Ansätze zur Umformung aufgezählt und die zugrunde liegenden Konzepte kurz und grob beschrieben.

Der folgende Abschnitt erklärt den Versuchsaufbau (XML-Testdokumente). Die verschiedenen Konversionsverfahren werden dann anhand des zuvor aufgestellten Anforderungskatalogs geprüft.

Anschließend werden die Ergebnisse der Tests vorgestellt und diskutiert. Dabei wird erörtert, ob und welche Verfahren die zuvor genannten Bedingungen erfüllen.

Falls keines der Konversionsverfahren den zuvor aufgestellten Kriterien genügt, folgt der Abschnitt, der einen eigenen Umwandlungsalgorithmus entwickelt[13].

Im Diskussions-Abschnitt werden die aktuellen Möglichkeiten zur Umwandlung von XML in JSON und wieder zurück abschließend eingeschätzt und bewertet. Dabei soll auch ein Ausblick auf weitere Verbesserungsmöglichkeiten oder alternative Wege zum Ziel erörtert werden.

Im letzten Kapitel zu verwandten Arbeiten werden weitere Arbeiten zum Thema beschrieben und Unterschiede zur eigenen Arbeit herausgearbeitet.

[13] Siehe Abschnitt III-B.

Literatur

[1] John Boyer. *Canonical XML Version 1.0*. W3C Recommendation. http://www.w3.org/TR/2001/REC-xml-c14n-20010315. W3C, 15. März 2001.

[2] John Boyer et al. "Experiences with JSON and XML Transformations". In: *W3C Workshop on Data and Services Integration*. 2011.

[3] Tim Bray. *The JavaScript Object Notation (JSON) Data Interchange Format*. RFC 7159. RFC Editor, März 2014.

[4] Adam DuVander. *JSON's Eight Year Convergence With XML*. ProgrammableWeb News. 26. Dez. 2013. URL: https://www.programmableweb.com/news/jsons-eight-year-convergence-xml/2013/12/26 (besucht am 06. 03. 2017).

[5] David Lee. "JXON: an architecture for schema and annotation driven JSON/XML bidirectional transformations". In: *Proceedings of Balisage: The Markup Conference*. 2011.

[6] Boci Lin et al. "Comparison between JSON and XML in Applications Based on AJAX". In: *2012 International Conference on Computer Science and Service System*. Aug. 2012, S. 1174–1177. DOI: 10.1109/CSSS.2012.297.

[7] Eve Maler et al. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. W3C Recommendation. http://www.w3.org/TR/2008/REC-xml-20081126/. W3C, 26. Nov. 2008.

[8] Jim Manico. *Handling Untrusted JSON Safely*. WhiteHat Security Blog. 11. Jan. 2012. URL: https://www.whitehatsec.com/blog/handling-untrusted-json-safely/ (besucht am 03. 03. 2017).

[9] Timothy D. Morgan und Omar Al Ibrahim. *XML Schema, DTD, and Entity Attacks: A Compendium of Known Techniques*. Techn. Ber. Virtual Security Research LLC, 19. Mai 2014.

[10] Nurzhan Nurseitov et al. "Comparison of JSON and XML Data Interchange Formats: A Case Study." In: *Caine* 2009 (2009), S. 157–162.

[11] Christopher Späth et al. "SoK: XML parser vulnerabilities". In: *10th USENIX Workshop on Offensive Technologies, WOOT 16, Austin, TX, August 8-9, 2016*. 2016.

[12] Gregory Steuck. *XXE (Xml eXternal Entity) Attack*. SecurityFocus Bugtraq Mailing list. 29. Okt. 2002. URL: http://www.securityfocus.com/archive/1/297714 (besucht am 03. 03. 2017).

[13] Guanhua Wang. "Improving Data Transmission in Web Applications via the Translation between XML and JSON". In: *2011 Third International Conference on Communications and Mobile Computing*. Apr. 2011, S. 182–185. DOI: 10.1109/CMC.2011.25.

[14] Peng Wang, Xiaodong Wu und Huamin Yang. "Analysis of the Efficiency of Data Transmission Format Based on Ajax Applications". In: *2011 International Conference of Information Technology, Computer Engineering and Management Sciences*. Bd. 4. Sep. 2011, S. 265–268. DOI: 10.1109/ICM.2011.199.

[15] Ming Ying und James Miller. "Refactoring legacy AJAX applications to improve the efficiency of the data exchange component". In: *Journal of Systems and Software* 86.1 (2013), S. 72–88. ISSN: 0164-1212. DOI: 10.1016/j.jss.2012.07.019.