JSON to Property graph mapping

Bacheloararbeit

Universität Rostock

Abdulrahman Kazzaz

Betreuer: Dr. -Ing. Holger Meyer
Dl Alf-Christian Schering

Agenda

1- Motivation

2- Problemstellung

3- Property Graph Model

4- JSON-Konzepte

5- Lösungsmöglichkeiten

6- Lösungsansatz

7- Zusammenfassung und Ausblick

1-Motivation

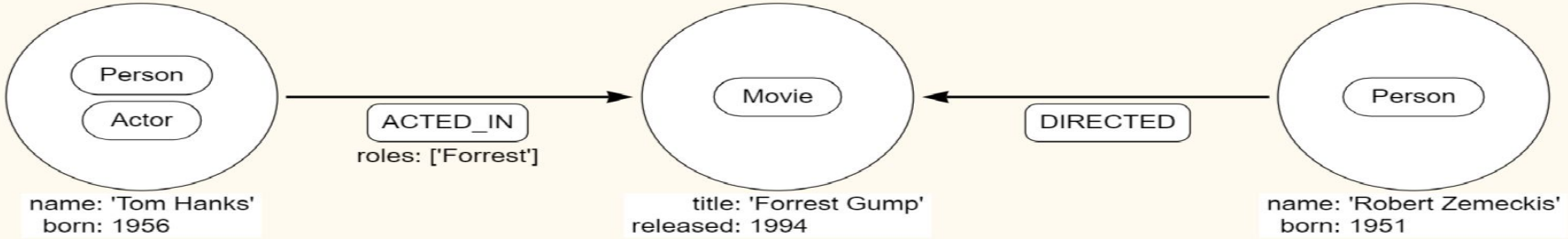
- in den letzten Jahren rasante Entwicklung in Graph-Gebiet.
- Ethnologen und Forscher*innen versuchen, Daten zu analysieren und durchzusuchen.
- Schwierigkeiten bei der Verarbeitung von Daten in verschiedenen Daten-Formate.
- Property Graph Modell bietet bessere Möglichkeiten, um Daten mit höher Genauigkeit zu analysieren.
- Auf den Graph-Daten erfolgt Graph Mining wie Community Detection, Graph Summarizing, frequent Subgraph Mining.

2-Problemstellung

- Entwicklung eines Tools zur Umwandlung von JSON-Daten in Property Graph-Daten.
- JSON-Dateien allgemein in Property Graphs umwandeln.
- Steuerung der Transformation durch benutzerdefinierte Regeln.
- Entwicklung einer Regel Spezifikationssprache auf der Grundlage der X2G-Sprache.
- Benutzer können Attribute und Inhalte aus JSON-Dateien auswählen und daraus Knoten, Kanten, Labels, Properties erzeugen.
- Benutzer können die Property Graph-Daten als CSV-Dateien herunterladen.
- CSV-Dateien können mit Graph-Programme wie Gephi visualisiert werden.

3-Property Graph Model

- Property Graph visualisiert Daten aus verschiedenen Daten-Formate.
- Property Graph besteht aus vier Konzepten: Knoten, Kanten, Labels und Properties.
- Sowohl Knoten als auch Kanten können Properties haben.
- Knoten und Kanten können null oder mehrere Labels haben.



4-JSON-Konzepte

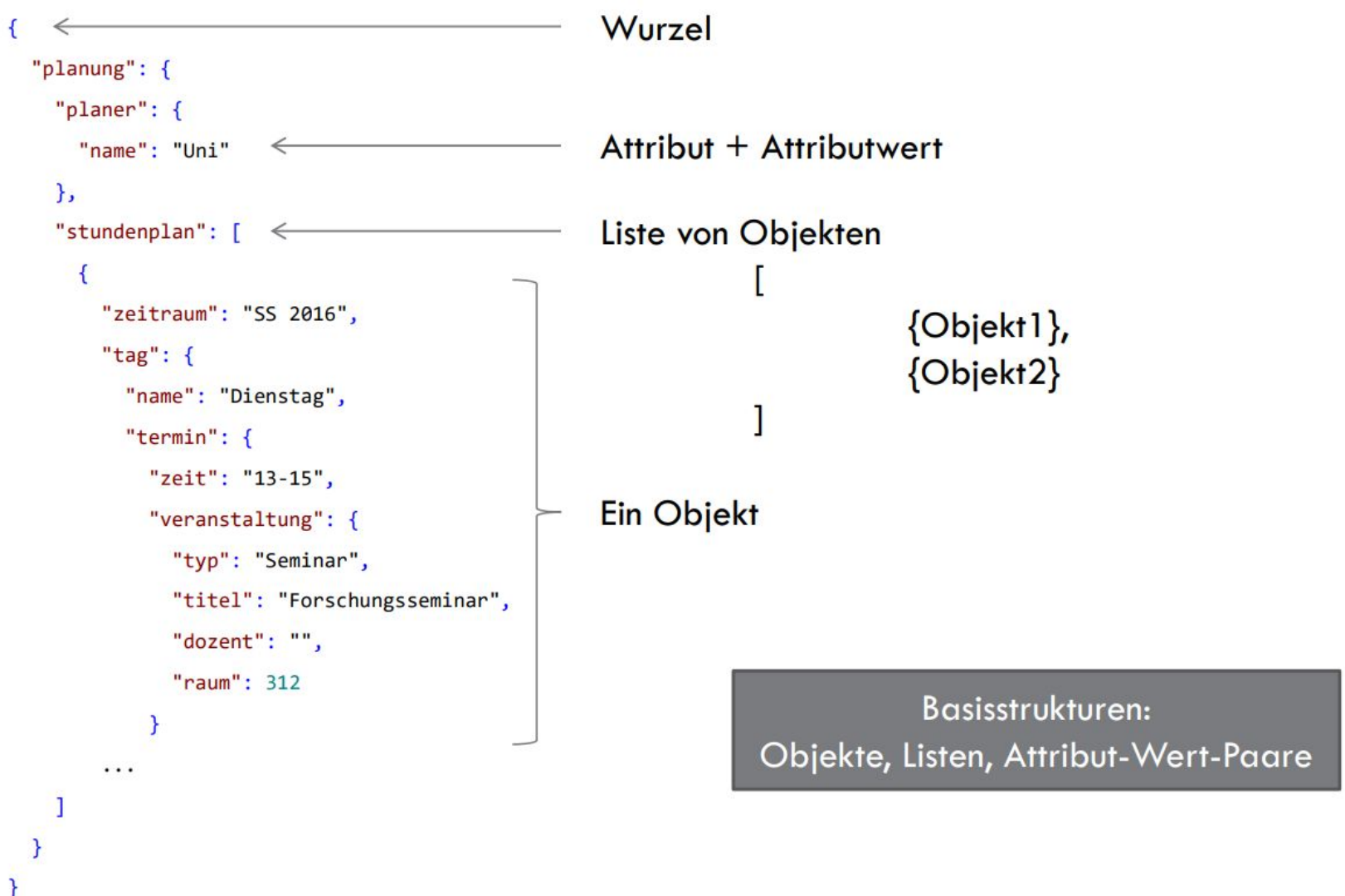
- Das JSON-Dokument besteht aus vier verschiedenen Konzepte: Wurzel, Attribute, Objekte und Arrays.

- Das JSON-Dokument beginnt mit einer Wurzel, der nicht unbedingt ein Element-Name enthalten muss.

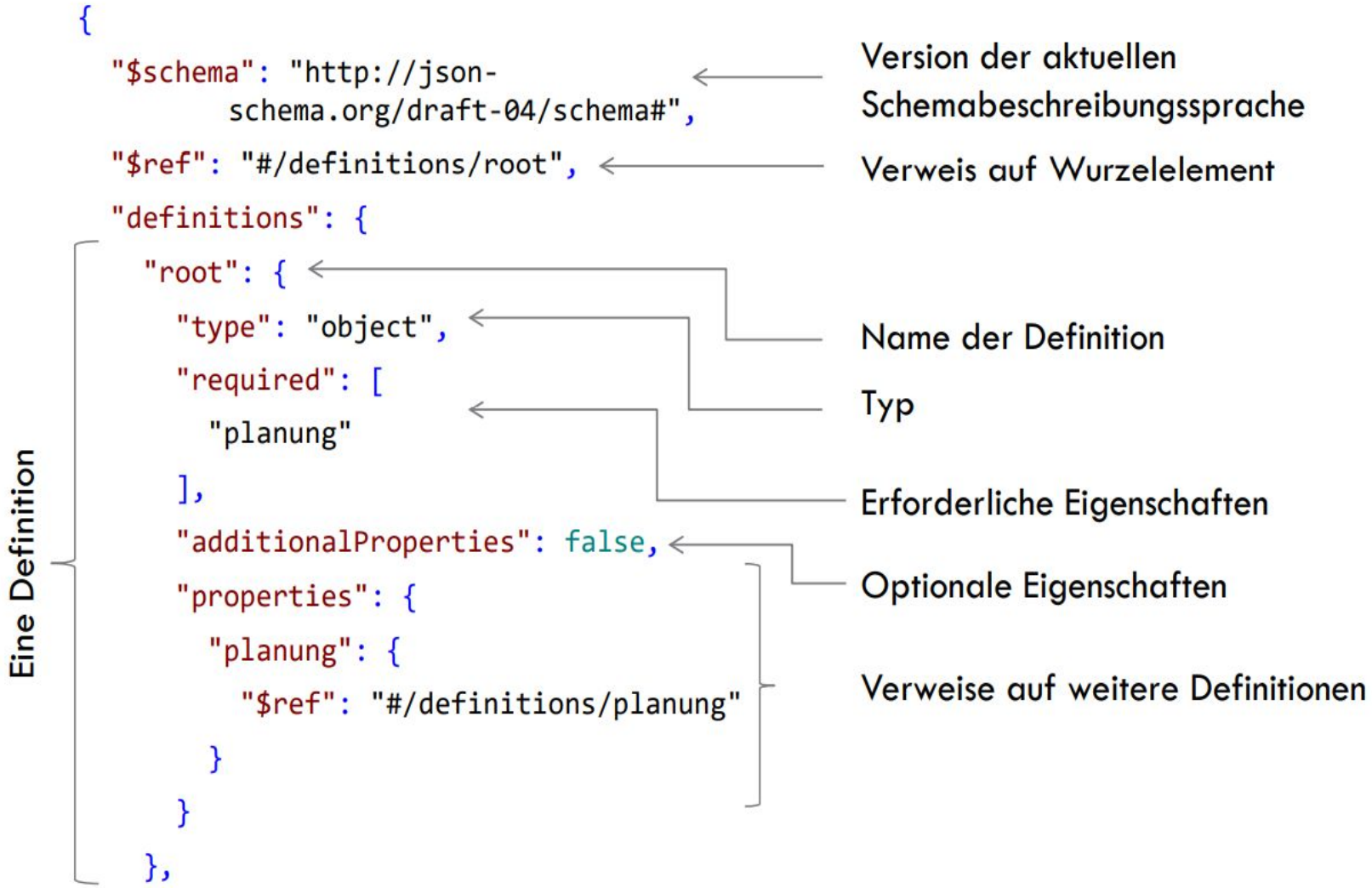
- JSON hat kein festes Schema.

- Jedes JSON-Dokument besteht aus zwei Strukturen: Menge von Name/Wert-Paare (Objekt) und Geordnete Liste von Werten.

- Die Werte können String, Number, Objekt, Array, True, False oder Null sein



Basisstrukturen:
Objekte, Listen, Attribut-Wert-Paare


{
"$schema": "http://json-schema.org/draft-04/schema#",
"$ref": "#/definitions/root",
"definitions": {
"root": {
"type": "object",
"required": [
"planung"
],
"additionalProperties": false,
"properties": {
"planung": {
"$ref": "#/definitions/planung"
}
}
},
Eine Definition
Version der aktuellen Schemabeschreibungssprache
Verweis auf Wurzelelement
Name der Definition
Typ
Erforderliche Eigenschaften
Optionale Eigenschaften
Verweise auf weitere Definitionen

5-Lösungsmöglichkeiten

- Ein Tool entwickeln, basierend auf JSONPath, um die JSON-Daten filtern zu können (JSONPath-Ansatz).

- JSON-Daten in andere Daten-Format umwandeln, für die bereits ein Tool zur Umwandlung in Property Graph existiert (JSON TO XML-Ansatz).

5.1-JSONPath-basierter Ansatz

- Bietet die selektive Extraktion von Daten aus JSON-Dokumenten an.
- JSONPath-Funktionen sind eingeschränkt.
- JSONPath-Funktionen erlauben keinen Zugriff auf Attributwerte.
- JSONPath ist kein Standard.

5.2-JSON TO XML Ansatz

- **Drei Möglichkeiten, um JSON zu XML zu transformieren:**

  1-XSLT 3.0

  2-XML.Serializer

  3-XML.toString

5.2.1-XSLT 3.0

- XSLT steht in den ersten Versionen für die Transformation von XML-Elemente zu anderen Daten-Formate wie zum Beispiel HTML.
- Das neuste Version XSLT 3.0 kann JSON-Dateien zu XML-Dateien transformieren.
- Der Installation vom XSLT-Prozessor ist erforderlich.
- JSON-Dateien müssen in einem XML-Tag gepackt werden.
- Eine Vorlage ist für die Umwandlung erforderlich.
- XSLT packt JSON-Daten in Map und Array Tags.
- ⇨ Die Extraktion von Daten mittels XPath ist schwierig, Wegen den Map und Array Tags.

```
<data>{
      "content": [
        {
          "id": 70805774,
          "value": "1001",
          "position": [1004.0,288.0,1050.0,324.0]
        }
      ]
}</data>
```

```
<?xml version="1.0"?>
<xsl:stylesheet
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:math="http://www.w3.org/2005/xpath-functions/math"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      exclude-result-prefixes="xs math" version="3.0">
      <xsl:output indent="yes" omit-xml-declaration="yes" />

      <xsl:template match="data">
          <xsl:copy-of select="json-to-xml(.)"/>
      </xsl:template>
</xsl:stylesheet>
```

```
<map xmlns="http://www.w3.org/2005/xpath-functions">
   <array key="content">
      <map>
         <number key="id">70805774</number>
         <string key="value">1001</string>
         <array key="position">
            <number>1004.0</number>
            <number>288.0</number>
            <number>1050.0</number>
            <number>324.0</number>
         </array>
      </map>
   </array>
</map>
```

5.2.3-XML.toString

- Hat einige Probleme:

  1-Das erzeugte XML-String hat kein Root-Element

  2-Achtet nicht auf der Reihenfolge der JSON-Datei.

- Transformiert JSON-Elemente ohne Manipulation.

- Hat einfache lösbare Probleme im Vergleich zu anderen Methoden.

- ⇨ Am besten geeignete Methode.

- JSONPath-Funktionen erlauben keinen Zugriff auf Attributwerte

  ⇨ Die Extraktion von Attribut-werte ist unmöglich.

- XML hat besseres Tool zur selektiven Extraktion von Daten (XPath).

- Die Extraktion von Attribut-werten mittels XPath ist möglich.

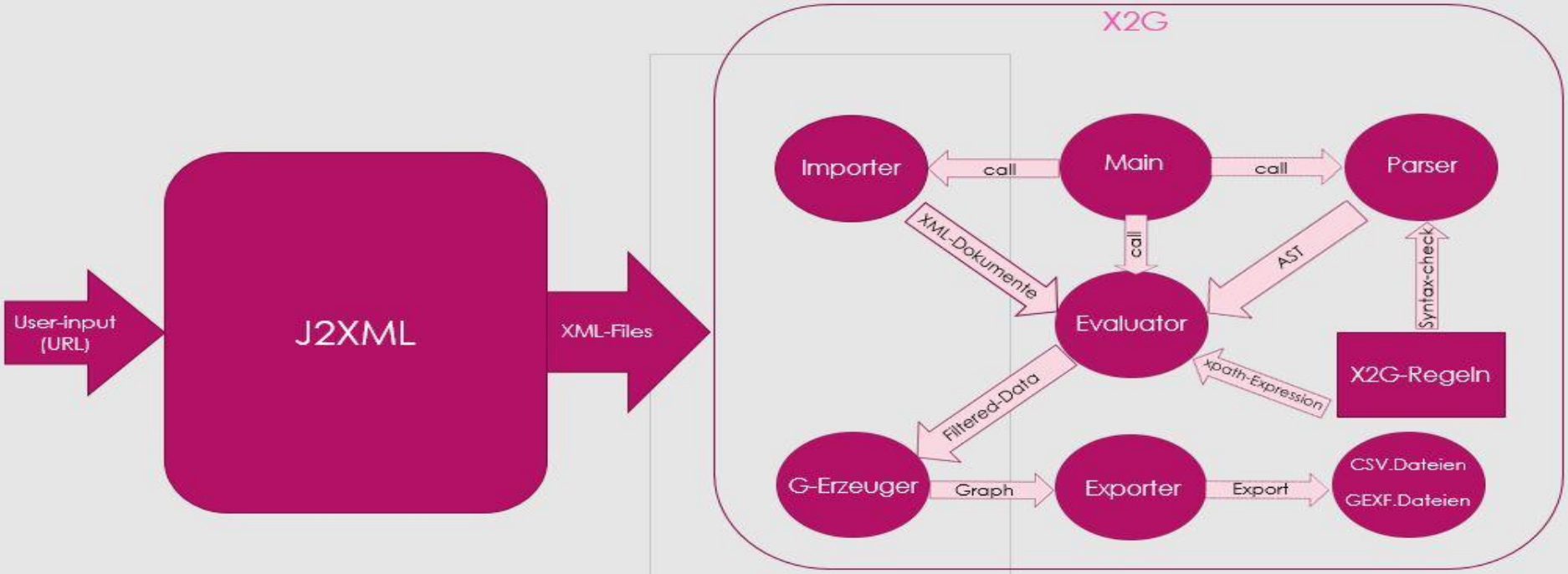
- Für XML existiert bereits ein Software (X2G) für die Umwandlung in Property Graph Model.

- Benutzer können in einem Tool JSON und XML Dokumente in Property Graph umwandeln.

- ⇨ JSON TO XML ist für diese Arbeit besser geeignet.

6-J2G

Besteht aus zwei Phasen (J2XML und X2G)

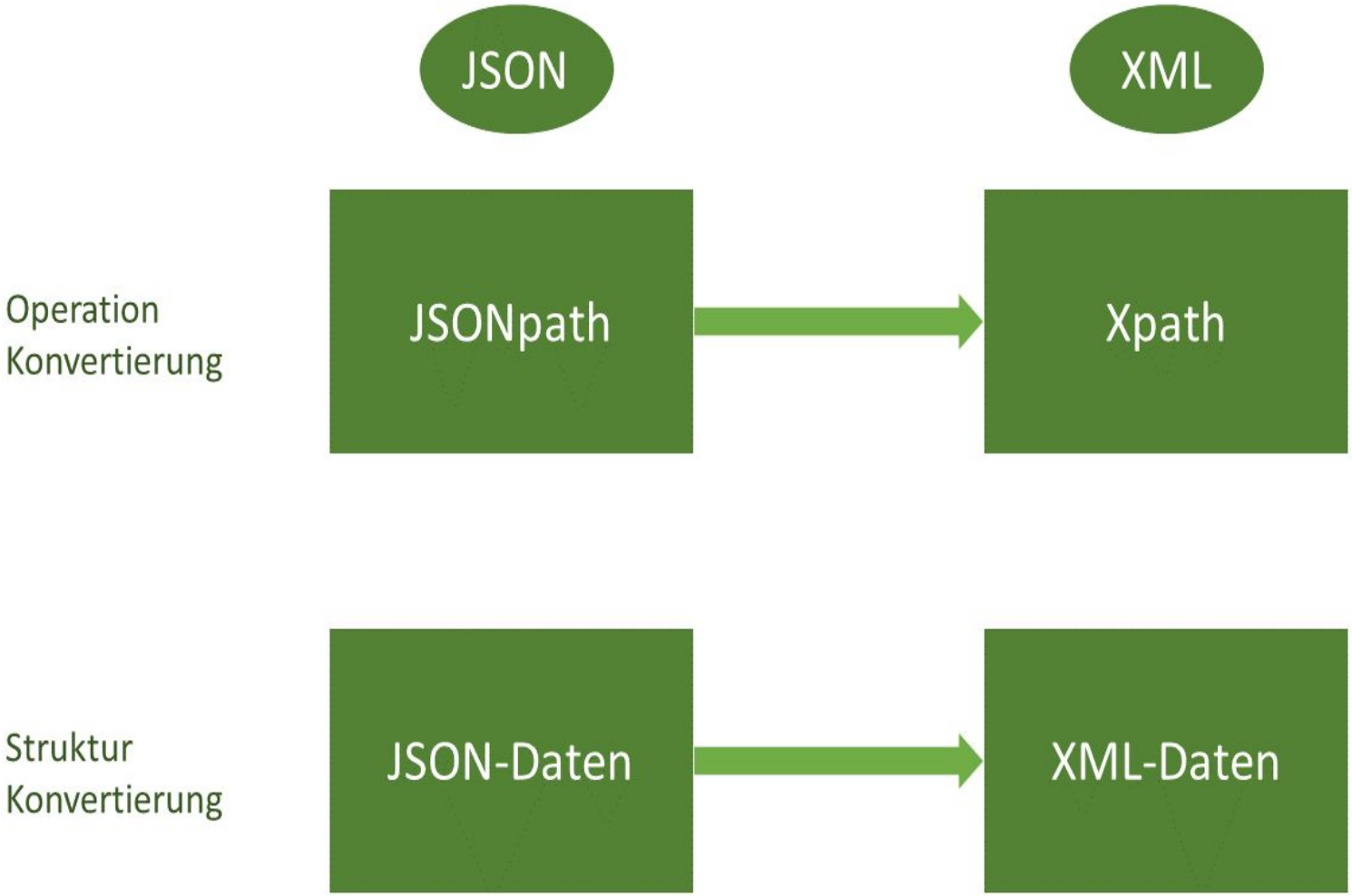


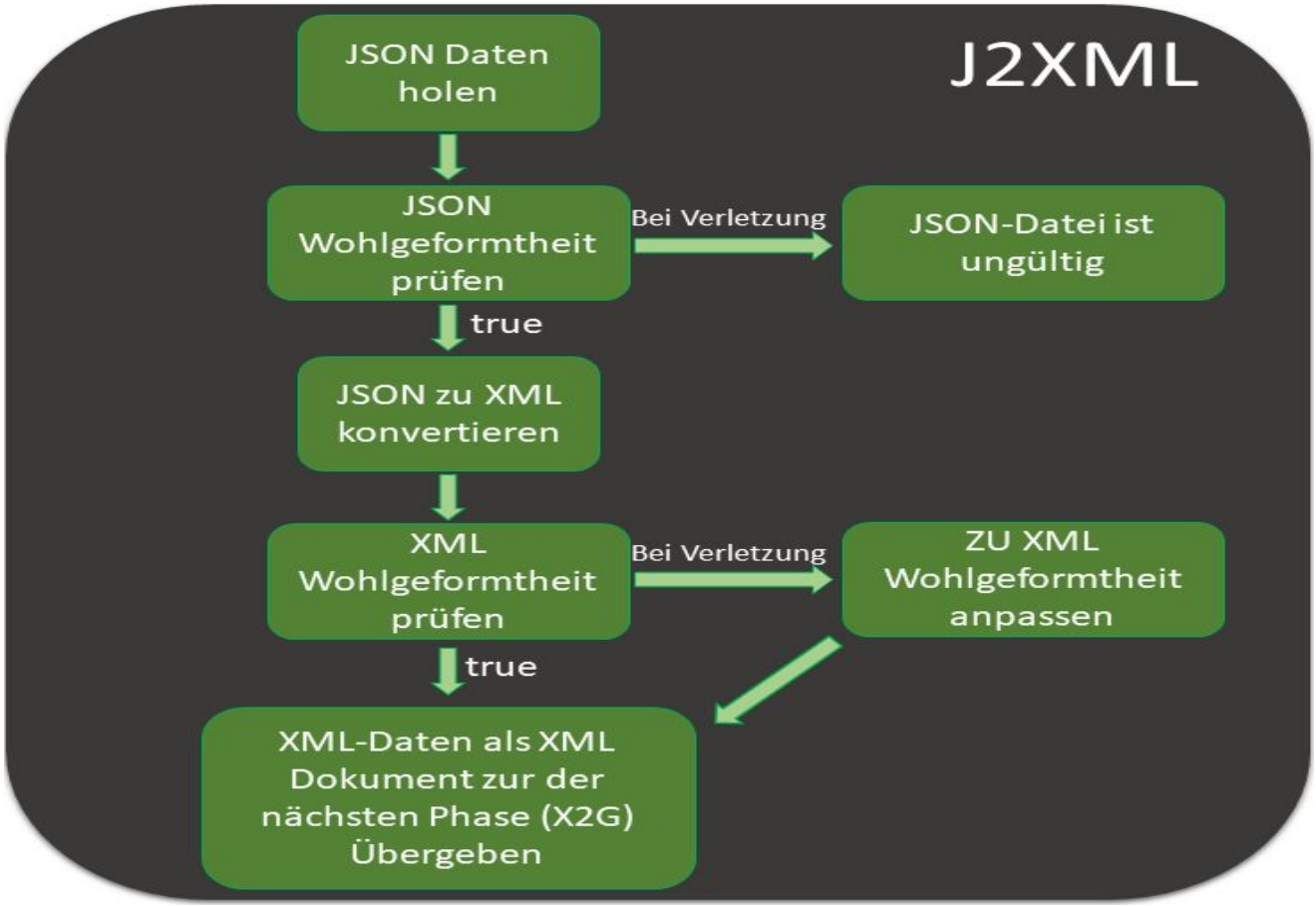
6.1-J2XML

- Transformiert JSON-Daten zu XML-Daten.

- Konvertiert JSONPath zu XPath.

- Leitet XML-Dokumente zu X2G weiter.


JSON
XML
Operation
Konvertierung
JSONpath
Xpath
Struktur
Konvertierung
JSON-Daten
XML-Daten


J2XML
JSON Daten holen
JSON Wohlgeformtheit prüfen
Bei Verletzung
JSON-Datei ist ungültig
true
JSON zu XML konvertieren
XML Wohlgeformtheit prüfen
Bei Verletzung
ZU XML Wohlgeformtheit anpassen
true
XML-Daten als XML Dokument zur der nächsten Phase (X2G) Übergeben

Konvertierung	
JSON Konzepte	XML Konzepte
Attribut	Element
Objekt	Element
Array	Elemente
Array ohne Name	Elemente mit dem Namen Array
Array-Element	Element mit dem Array-Namen

Wohlgeformtheit	
JSON	XML
Das Root-Element wird durch eine geschweifte Klammer oder eine Ecke Klammer gekennzeichnet	Das XML-Dokument darf nur ein Root-Element enthalten.
Das Root-Element kann keinen Elementnamen enthalten	Das Root-Element muss einen Elementnamen enthalten
Eine Datei beginnt mit einer geschweiften Klammer oder mit einer Ecke Klammer und endet mit deren Schließung	Eine Datei beginnt mit dem öffnenden Root-Element Tag und endet mit dessen schließendem Tag
Ein Objekt beginnt mit einer geschweiften Klammer und endet mit deren Schließung	Ein Element beginnt mit öffnendes Tag
Eine Liste beginnt mit einer Ecke Klammer und endet mit deren Schließung	Ein Element endet mit schließendes Tag
Ein Objekt oder eine Liste darf beliebige Zeichen enthalten	Ein Element darf keine Sonderzeichen wie: (;) und (@) u.s.w enthalten
Der Bezeichner (Name) darf mit einer Zahl oder mit Sonderzeichen beginnen	Der Bezeichner (Name) darf nicht mit einer Zahl oder mit Sonderzeichen beginnen
Ein leeres Objekt oder eine leere Liste werden durch zwei leerende geschweiften oder Ecke klammer gekennzeichnet	Ein leeres Element wird durch leerendes öffnendes und schließendes Tags gekennzeichnet

6.1.4-XML-Wohlgeformtheit anpassen

- Fügt das Root-Element hinzu.

- Fügt das Zeichen (E-) vor Elementnamen, die mit einer natürlichen Zahl beginnen.

- Fügt das Zeichen (E-) vor Elementnamen, die mit einem Sonderzeichen wie (@, :) beginnen.

- Das Zeichen (E-) ist die Abkürzung für Element.

```
{
"1planung": {
"planer": {
"name": "Uni"
},
"stundenplan": [
{
"zeitraum": "SS 2016",
"tag": {
"name": "Dienstag",
"termin": {
"zeit": "13-15",
"veranstaltung": {
"typ": "Seminar",
"titel": "Forschungsseminar",
"dozent": "",
"raum": "312"}}}
}]
}
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
<1planung>
<planer>
<name>Uni</name>
</planer>
<stundenplan>
<zeitraum>SS 2016</zeitraum>
<tag>
<termin>
<veranstaltung>
<titel>Forschungsseminar</titel>
<raum>312</raum>
<dozent/>
<typ>Seminar</typ>
</veranstaltung>
<zeit>13-15</zeit>
</termin>
<name>Dienstag</name>
</tag>
</stundenplan>
</1planung>
</root>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
<E-1planung>
<planer>
<name>Uni</name>
</planer>
<stundenplan>
<zeitraum>SS 2016</zeitraum>
<tag>
<termin>
<veranstaltung>
<titel>Forschungsseminar</titel>
<raum>312</raum>
<dozent/>
<typ>Seminar</typ>
</veranstaltung>
<zeit>13-15</zeit>
</termin>
<name>Dienstag</name>
</tag>
</stundenplan>
</E-1planung>
</root>
```

6.2-Operationelle Konvertierung

- Konvertiert JSONPath-Ausdrücke zu XPath-Ausdrücke.

- Die Konvertierung basiert auf eine Regelsprache.

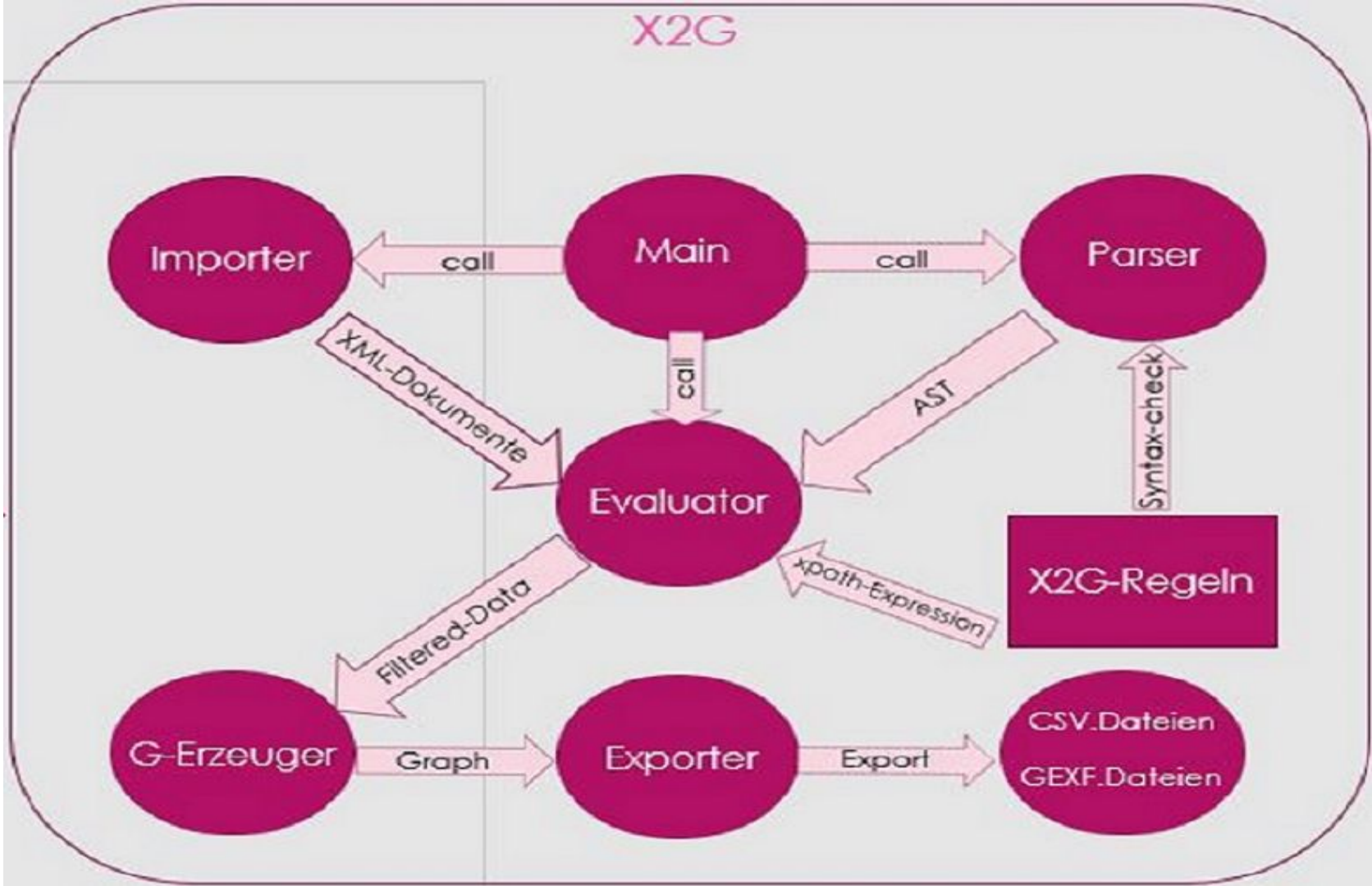
- Achtet bei der Konvertierung auf die XPath-Semantik.

Konvertierung		
JSONPath	XPath	Beschreibung
§	/root	das Stammobjekt/-element
@	.	das aktuelle Objekt/Element
.	/	das nächste Objekt/Element
==	=	ist gleich Symbol
-1:	last()	das letzte Objekt/Element
-n:	$position() < last() - n$	die letzten n-Objekte/Elemente
:n	$position() < n + 1$	die ersten n-Objekte/Elemente
&&	and	Logisches und Symbol
‖	or	Logisches oder Symbol
..	//	rekursiver Abstieg
[ ]	[ ]	Subscript-Operator
n	n+1	das n-ten Objekt/Element
? ()	[ ]	wendet einen Filterausdruck an.

JSONPath	XPath	Beschreibung
§.store.book[*].author	/root/store/book/author	the authors of all books in the store
§..author	/root//author	all authors
§.store.*	/root/store/*	all things in store, which are some books and a red bicycle.
§.store..price	/root/store//price	the price of everything in the store
§..book[2]	/root//book[3]	the third book
§..book[(@.length-1)] §..book[-1:]	/root//book[last()]	the last book in order.
§..book[0,1] §..book[:2]	/root//book[$position() < 3$]	the first two books
§..book[?(@.isbn)]	/root//book[isbn]	filter all books with isbn number
§..book[?(@.*price* < 10)]	/root//book[*price* < 10]	filter all books cheapier than 10
§..*	/root//*	all Elements in XML document. All members of JSON structure.

6.2X2G

- Importiert XML-Files.
- Basiert auf eine Regel-Spezifikationssprache.
- Wandelt XML-Daten in Property Graph Daten um.
- Exportiert das Property Graph als CSV-Dateien


X2G
Importer
call
Main
call
Parser
XML-Dokumente
call
AST
Syntax-check
Evaluator
X2G-Regeln
xpath-Expression
Filtered-Data
G-Erzeuger
Graph
Exporter
Export
CSV.Dateien
GEXF.Dateien

6.2.3Regelsprache

Die Regelsprache weist die folgenden Hauptkonzepte:

- Extrahieren von XML-Fragmenten nach XPath-Ausdruck, JSON-Fragmenten nach JSON-Pfad und relationalen Daten nach SQL-Abfragen.
- Extrahierte Daten werden an Variablen gebunden.
- Generierung von Knoten und Kanten aus Literalen und Auswertung von Variablenbindungen.
- Verschachtelte Auswertung von Anweisungen im Kontext anderer Anweisungen und Variablenbindungen.
- Regeln haben einen Kopf, der einigen XPath-Ausdrücken oder Knoten- oder Kantenbeschriftungen entspricht. Der Text kann lokale Übereinstimmungs-Regeln oder Knoten- und Kantengenerierung-Anweisungen enthalten.
- Regeln auf der obersten Ebene werden in der Reihenfolge des Dokuments ausgewertet.
- Verschachtelte Regeln werden im Kontext der Regel ausgewertet, die sie enthält.
- Es gibt keine Ausnahmebehandlung durch Design.

```
// nested match example, the second match is evaluated within the context of the first
match xpath("//cd") using $c {
      create node $cn label "cd" {
            // properties
            title = $c.xpath("title/text()"),
            src = $c.xpath("src/text()"),
            avail = true,
            unique (title)
      },
      match $c.xpath("artist") using $a {
            create node $an label "artist" {
                  name = $a.xpath("text()"),
                  artist = $cn.src,
                  unique (name)
            },
            create edge $e from $an to $cn label "interpret" {
                  alt = "disc-artist"
            }
      }
}
```

```
// nested match example, the second match is evaluated within the context of the first
match jpath("$..cd") using $c {
      create node $cn label "cd" {
            // properties
            title = $c.jpath("$..title.text()"),
            src = $c.jpath("$..src.text()"),
            avail = true,
            unique (title)
      },
      match $c.jpath("artist") using $a {
            create node $an label "artist" {
                  name = $a.jpath("text()"),
                  artist = $cn.src,
                  unique (name)
            },
            create edge $e from $an to $cn label "interpret" {
                  alt = "disc-artist"
            }
      }
}
```

```
id,label,properties
5,cd,avail,true,src,http://www.stoatmusic.com/,title,Future come and get me
9,cd,avail,true,src,http://www.led-zeppelin.com/,title,"IV, Four Symbols"
1,cd,avail,true,src,http://www.led-zeppelin.com/,title,The Song Remains the Same
3,cd,avail,true,src,http://www.sufjan.com/,title,Illinois
7,cd,avail,true,src,http://www.whitestripes.com/,title,Get behind me satan
4,artist,artist,http://www.sufjan.com/,name,Sufjan Stevens
8,artist,artist,http://www.whitestripes.com/,name,The White Stripes
2,artist,artist,http://www.led-zeppelin.com/,name,Led Zeppelin
6,artist,artist,http://www.stoatmusic.com/,name,Stoat
```

```
src-id,dst-id,label,properties
2,9,interpret,alt,disc-artist
6,5,interpret,alt,disc-artist
8,7,interpret,alt,disc-artist
4,3,interpret,alt,disc-artist
2,1,interpret,alt,disc-artist
```

7-Zusammenfassung und Ausblick

- Konzept/Entwurf erstellt.
- Folgende Teile wurden implementiert:

  J2XML Phase und das enthält:

  -JSON-XML Konverter (Strukturelle Konvertierung).

  -JSONPath-XPath Konverter.

- Folgende Teile wurden nicht implementiert:

  -Ausgabe des generierten Graphs (GEXF-Format).

- was zu ergänzen ist:

  -J2XML mit X2G einbinden.

- Mögliche /zukünftige Erweiterungen:

  -Key-Values Stores

  Daraus Graphen erzeugen.

Vielen Dank für Ihre Aufmerksamkeit

5.2.2-XML.Serializer

- XML.Serializer ist eine vordefinierte Klasse in Java.
- Erzeugt das Element (o) als Wurzelelement
- ⇨ was zu Problemen mit XPath führt.
- Wandelt JSON-Listen als Objekten.

```
{"store": {
  "book":   [
        {
      "category": "reference",
      "author": "Nigel Rees",
      "title": "Sayings of the Century",
      "price": 8.95
    },
        {
      "category": "fiction",
      "author": "Evelyn Waugh",
      "title": "Sword of Honour",
      "price": 12.99
    },
        {
      "category": "fiction",
      "author": "Herman Melville",
      "title": "Moby Dick",
      "isbn": "0-553-21311-3",
      "price": 8.99
    },
        {
      "category": "fiction",
      "author": "J. R. R. Tolkien",
      "title": "The Lord of the Rings",
      "isbn": "0-395-19395-8",
      "price": 22.99
    }
  ],
  "bicycle":   {
    "color": "red",
    "price": 19.95
  }
}}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<o>
  <store class="object">
    <bicycle class="object">
      <color type="string">red</color>
      <price type="number">19.95</price>
    </bicycle>
    <book class="array">
      <e class="object">
        <author type="string">Nigel Rees</author>
        <category type="string">reference</category>
        <price type="number">8.95</price>
        <title type="string">Sayings of the Century</title>
      </e>
      <e class="object">
        <author type="string">Evelyn Waugh</author>
        <category type="string">fiction</category>
        <price type="number">12.99</price>
        <title type="string">Sword of Honour</title>
      </e>
      <e class="object">
        <author type="string">Herman Melville</author>
        <category type="string">fiction</category>
        <isbn type="string">0-553-21311-3</isbn>
        <price type="number">8.99</price>
        <title type="string">Moby Dick</title>
      </e>
      <e class="object">
        <author type="string">J. R. R. Tolkien</author>
        <category type="string">fiction</category>
        <isbn type="string">0-395-19395-8</isbn>
        <price type="number">22.99</price>
        <title type="string">The Lord of the Rings</title>
      </e>
    </book>
  </store>
</o>
```