

XML und JSON Parsing am Beispiel

Auf dieser Seite sammeln wir Beispiele, wie man XML oder JSON Daten mit Synesty einlesen kann. Diese Seiten sind Ergänzung zu [XML2Spreadsheet](#) und [JSON2Spreadsheet](#).

- 1 [Beispiel 1 - XML - Einfache XML Struktur](#)
- 2 [Beispiel 2 - XML - Komplexe Struktur mit Attributen und Referenzelementen](#)
- 3 [Beispiel 3 - XML Produktkatalog mit mehreren Merkmal Tags](#)
- 4 [Beispiel 4 - SOAP XML-Webservice Response verarbeiten](#)
- 5 [Beispiel 5 - JSON Parsing](#)
- 6 [Beispiel 6 - JSON Parsing](#)
- 7 [Mehr zum Thema XML, JSON und API-Anbindungen](#)

Beispiel 1 - XML - Einfache XML Struktur

Quelle: [XML2Spreadsheet#XMLParsingVariante1](#)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ARTIKELLISTE>
  <ARTIKEL>
    <LFSN>ZEG</LFSN>
    <LFKURZBEZ>ZEG</LFKURZBEZ>
    <INTARTNR>8592</INTARTNR>
    <ARTNR>010-12983</ARTNR>
    <REFARTNR>010-12983</REFARTNR>
    <HERSTNR>870310 / 10870310</HERSTNR>
    <MARKE>Schwalbe</MARKE>
    <MARKENLOGO></MARKENLOGO>
    <MODELL>High Pressure</MODELL>
    <MODELLJAHR>0</MODELLJAHR>
    <FARBE>blau</FARBE>
  </ARTIKEL>
  <ARTIKEL>
    <LFSN>ZEG</LFSN>
    <LFKURZBEZ>ZEG</LFKURZBEZ>
    <INTARTNR>8593</INTARTNR>
    <ARTNR>010-12984</ARTNR>
    <REFARTNR>010-12984</REFARTNR>
    <HERSTNR>870311 / 10870311</HERSTNR>
    <MARKE>Simson</MARKE>
    <MARKENLOGO></MARKENLOGO>
    <MODELL>High Pressure</MODELL>
    <MODELLJAHR>0</MODELLJAHR>
    <FARBE>rot</FARBE>
  </ARTIKEL>
</ARTIKELLISTE>
```

Parsing Template:

```
<#assign row = target.addRow()>

<#list xml["ARTIKELLISTE"]["ARTIKEL"] as art>
  <#assign row = target.addRow()>
  ${addColumn(row, art)}
</#list>
```

oder Alternative:

```

<#assign row = target.addRow()

<#list xml["ARTIKELLISTE"]["ARTIKEL"] as art>
  <#assign row = target.addRow()
  ${row.addCol("Artikelnummer",art["ARTNR"])}
  ${row.addCol("Marke",art["MARKE"])}
  ${row.addCol("Modell",art["MODELL"])}
</#list>

```

Ergebnis:

Artikelnummer	Marke	Modell
010-12983	Schwalbe	High Pressure
010-12984	Simson	High Pressure

Beispiel 2 - XML - Komplexe Struktur mit Attributen und Referenzelementen

Anforderung:

Ich habe eine Artikeldatei, die in einem XML Abschnitt Artikel mit Varianten enthält. Nach diesem Schema sind die Artikel mit Eigenschaften der Varianten in Unterabschnitten enthalten.

Es soll für jeden GrIndex eine Zeile geschrieben werden.

```

<Artikeldaten>
  <Artikel Aktiv="1">
    <ArtikelNr>123456</ArtikelNr>
    <ArtikelName>Toller Artikel</ArtikelName>
  <VKPreise>
    <VKPreis GrIndex="6" Waehrung="EUR" VK1="34,95" VK2="0,00" VK3="0,00">34,95</VKPreis>
    <VKPreis GrIndex="7" Waehrung="EUR" VK1="34,95" VK2="0,00" VK3="0,00">34,95</VKPreis>
  </VKPreise>
  <Bestaende>
    <Bestand GrIndex="6">2</Bestand>
    <Bestand GrIndex="7">8</Bestand>
  </Bestaende>
  <BestaendeDetails>
    <BestaendeDetail Filiale="001" GLN="4399902192156">
      <Bestand GrIndex="6">1</Bestand>
      <Bestand GrIndex="7">1</Bestand>
    </BestaendeDetail>
    <BestaendeDetail Filiale="002" GLN="4399902347143">
      <Bestand GrIndex="6">1</Bestand>
      <Bestand GrIndex="7">1</Bestand>
    </BestaendeDetail>
    <BestaendeDetail Filiale="003" GLN="4399902347136">
      <Bestand GrIndex="6">0</Bestand>
      <Bestand GrIndex="7">6</Bestand>
    </BestaendeDetail>
  </BestaendeDetails>
  <EAN_Codes>
    <EAN GrIndex="6" />
    <EAN GrIndex="7">8023121212111</EAN>
  </EAN_Codes>
</Artikel>
</Artikeldaten>

```

Parsing Template:

```

<#assign row = target.addRow()
<#list xml["Artikeldaten"]["Artikel"] as art>
  <#assign row = target.addRow()
  ${addColumn(row, art)}
  <#list art['VKPreise']['VKPreis'] as vkp>
    <#assign vkprow = target.addRow()
    ${vkprow.addCol("VKPreis", vkp)}
    ${vkprow.addCol("GrIndex", attr("GrIndex", vkp)!)}
    ${vkprow.addCol("VK1", attr("VK1", vkp)!)}

    <!-- now the difficult part: getting the stock and EAN for each GrIndex -->
    ${vkprow.addCol("Bestand", art['Bestaende']['Bestand']?filter(bestand -> attr("GrIndex", bestand)! == attr("GrIndex", vkp)!)?first!)}
    ${vkprow.addCol("EANCode", art['EAN_Codes']['EAN']?filter(ean -> attr("GrIndex", ean)! == attr("GrIndex", vkp)!)?first!)}

    <#list art["BestaendeDetails"]["BestaendeDetail"] as bestandDetail>
      <#list bestandDetail["Bestand"] as bestand>
        <#if attr("GrIndex", bestand)! == attr("GrIndex", vkp)>
          ${vkprow.addCol("Bestand_Filiale_" + attr("Filiale", bestandDetail), bestand!)}
        </#if>
      </#list>
    </#list>
  </#list>
</#list>

```

Ergebnis

Vorschau des Steps

×

Cache aktiv [Zurücksetzen](#)

[Gekürzte Spalten erweitern](#)

Artikel_BestaendeDetails	Artikel_EAN_Codes	VKPreis	Grindex	VK1	Bestand	EANCode	Bestand_Filiale_001	Bestand_Filiale_002	Bestand_Filiale_003
<BestaendeDetails> <BestaendeDetail Filiale="001" GLN="4399902192156"> <Bestand GrIndex="6">1</Bestand> <Bestand GrIndex="7">1</Bestand> (...)	<EAN_Codes> <EAN Grindex="6"/> <EAN Grindex="7">8023121212111</EAN> </EAN_Codes>								
		34,95	6	34,95	2		1	1	0
		34,95	7	34,95	8	8023121212111	1	1	6

Weitere Informationen

- Wichtigste Freemarker-Funktion: [?filter](#)
- Zugriff auf XML Attribute per [attr\(attributenname, element\) Funktion](#)
- Da die Struktur etwas komplexer ist muss man auf Varianteebene auf [\\${row.addCol\(\)}](#) Syntax umsteigen, da dort die [\\${addColumn\(\)}](#)-Magic an ihre Grenzen stößt
- Ggf. noch mit [exclude](#) nicht benötigte Spalten entfernen (oder nur mit include arbeiten).

Beispiel 3 - XML Produktkatalog mit mehreren Merkmal Tags

Quelle: <https://support.synesty.com/de/support/discussions/topics/11000034935>

Anforderung: "Mein Problem ist, dass der Knoten "Merkmal" mehrfach vorkommen kann und ich insgesamt ca. 500 verschiedene Merkmalschlüssel habe."

```
<?xml version="1.0" encoding="UTF-8"?>
<ZEGSHOP>
  <DELETE>1</DELETE>
  <HAUPTKATEGORIE>
    <KATEGORIE>
      <ARTIKEL>
        <INTARTNR>22581</INTARTNR>
        <MERKMAL>
          <MERKMALSCHLUESSEL>FARB</MERKMALSCHLUESSEL>
          <AUSPRAEGUNGSSCHLUESSEL>SW</AUSPRAEGUNGSSCHLUESSEL>
          <MERKMAL>Farbe</MERKMAL>
          <AUSPRAEGUNG>schwarz</AUSPRAEGUNG>
        </MERKMAL>
        <MERKMAL>
          <MERKMALSCHLUESSEL>FARR</MERKMALSCHLUESSEL>
          <AUSPRAEGUNGSSCHLUESSEL>SW</AUSPRAEGUNGSSCHLUESSEL>
          <MERKMAL>Farbrichtung</MERKMAL>
          <AUSPRAEGUNG>schwarz</AUSPRAEGUNG>
        </MERKMAL>
        <MERKMAL>
          <MERKMALSCHLUESSEL>FUTI</MERKMALSCHLUESSEL>
          <AUSPRAEGUNGSSCHLUESSEL>FU200</AUSPRAEGUNGSSCHLUESSEL>
          <MERKMAL>Funktion</MERKMAL>
          <AUSPRAEGUNG>31</AUSPRAEGUNG>
        </MERKMAL>
      </ARTIKEL>
    </KATEGORIE>
  </HAUPTKATEGORIE>
</ZEGSHOP>
```

Parsing Code:

```
<#assign row = target.addRow()>
<#list xml["ZEGSHOP"]["HAUPTKATEGORIE"]["KATEGORIE"]["ARTIKEL"] as p>
  <#assign row = target.addRow()>
  <!-- ${addColumns(row, p, '', {'columns':['MERKMAL'], 'mode':'exclude'})} -->

  <#list p["MERKMAL"] as t>
    ${row.addCol("Merkmalschlüssel-" + t["MERKMALSCHLUESSEL"], t["MERKMALSCHLUESSEL"])}
    ${row.addCol("Ausprägungsschlüssel-" + t["MERKMALSCHLUESSEL"], t["AUSPRAEGUNGSSCHLUESSEL"])}
    ${row.addCol("Merkmal-" + t["MERKMALSCHLUESSEL"], t["MERKMAL"])}
    ${row.addCol("Ausprägung-" + t["MERKMALSCHLUESSEL"], t["AUSPRAEGUNG"])}
  </#list>

  ${addColumns(row, p, '', {'columns':['MERKMAL'], 'mode':'exclude'})}

</#list>
```

Ergebnis:

XMLReader

Vorschau erfolgreich ausgeführt

Die Ausgabe dient zu Vorschauzwecken und kann von der Ausgabe der echten Ausführung abweichen.

SPREADSHEET output@XML2Spreadsheet_1

Cache aktiv Zurücksetzen

Merkmalsschlüssel-FARB	Ausprägungsschlüssel-FARB	Merkmal-FARB	Ausprägung-FARB	Merkmalsschlüssel-FARR	Ausprägungsschlüssel-FARR	Merkmal-FARR	Ausprägung-FARR	Merkmalsschlüssel-FUTI	Ausprägungsschlüssel-FUTI	Merkmal-FUTI	Ausprägung-FUTI	Merkmalsschlüssel-GEW
FARB	SW	Farbe	schwarz	FARR	SW	Farbrichtung	schwarz	FUTI	FUJ200	Funktion	31	GEW

Weitere Informationen

- Herausforderung ist, dass die Merkmals-Werte in Tags wie z.B. <MERKMALSCHLUESSEL> , gleichzeitig auch die Spaltentitel sein sollen
- allein mit `addColumn()` kommt man nicht weit, da die Anforderung zu speziell ist
- daher wird die Funktion `row.addColumn(title, value)` genutzt (siehe [XML2Spreadsheet - Parsing Variante 2](#)), die volle Kontrolle über die Spaltentitel bietet

Beispiel 4 - SOAP XML-Webservice Response verarbeiten

SOAP Webservices sind im Grund nichts anderes als HTTP-Requests mit XML als Austauschformat analog zu den mittlerweile weit verbreiteten [REST-APIs](#).

D.h. man arbeitet genau so mit den Steps wie [APICall](#), [SpreadsheetURLDownload](#) und [XMLReader](#).

Beim Verarbeiten der etwas komplexeren XML-Struktur muss man ein paar Dinge beachten wie z.B. XML-Namespaces (die `xmlns` Angaben im Beispiel unten).

Das Beispiel zeigt eine beispielhafte SOAP XML-Response eines Webservices. Darin enthalten ist eine Liste mit zwei **PackageGroupRequest-Objekten**, die als Spreadsheet ausgegeben werden sollen.

```

<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns="http://www.meinprefix.net/foo/2013/04">
<soap:Header/>
  <soap:Body>

    <PackageGroupRequest
      xmlns="http://www.meinprefix.net/foo/2013/04"
      AuthKey="XXXXX">
      <TravelPeriod>
        <DepartureDate>2015-08-01</DepartureDate>
        <Duration>128</Duration>
      </TravelPeriod>
      <Travellers>
        <Adult Age="28" />
      </Travellers>
      <Hotel>
        <Rooms />
      </Hotel>
    </PackageGroupRequest>

    <PackageGroupRequest
      xmlns="http://www.meinprefix.net/foo/2013/04"
      AuthKey="XXXXX">
      <TravelPeriod>
        <DepartureDate>2013-08-01</DepartureDate>
        <Duration>31</Duration>
      </TravelPeriod>
      <Travellers>
        <Adult Age="19" />
      </Travellers>
      <Hotel>
        <Rooms />
      </Hotel>
    </PackageGroupRequest>

  </soap:Body>
</soap:Envelope>

```

parsingTemplate im XMLReader:

```

<#ftl ns_prefixes={"A":"http://www.w3.org/2003/05/soap-envelope",
                  "B":"http://www.meinprefix.net/foo/2013/04"}>

<#assign row = target.addRow()>
<#list xml["A:Envelope"]["A:Body"]["B:PackageGroupRequest"] as item>
  <#assign row = target.addRow()>
  ${addColumn(row, item, "", {"autoExpand": "asColumns"})}
</#list>

```

Ergebnis:

XMLReader SOAP Response-mit-Namespace

Vorschau erfolgreich ausgeführt

Die Ausgabe dient zu Vorschauzwecken und kann von der Ausgabe der echten Ausführung abweichen.

SPREADSHEET output@XML2Spreadsheet_7

Cache aktiv  Zurücksetzen[Download Preview CSV](#) [Download CSV](#)

Vorschau Filter



Filter anwenden

[Gekürzte Spalten erweitern](#)

TravelPeriod_DepartureDate	TravelPeriod_Duration	Travellers_Adult	Hotel_Rooms
2015-08-01	128		
2013-08-01	31		

Weitere Informationen und Besonderheiten

- wichtig ist die Angaben der Namespaces in ns_prefixes mit den Aliassen bzw. Prefixen A und B. Dieses A und B wird auch in der <#list> Anweisung verwendet
- {"autoExpand": "asColumns"} ist eine Option des XMLReader, um Unter-Nodes eines XML-Tags automatisch als Spalten darzustellen.

Beispiel 5 - JSON Parsing

Quelle: <https://forum.synesty.com/t/json-parsingtemplate/2310>**Anforderung:**

Im Spreadsheet werden die Werte

- producent_4711
- Produzent 4711
- 123456
- Stadt
- Land

jeweils in einer Spalte benötigt.

JSON:

```

{
  "_links": {
    "self": {
      "href": "xxxxx"
    },
    "first": {
      "href": "xxxxx"
    },
    "next": {
      "href": "xxxxx"
    }
  },
  "_embedded": {
    "items": [
      {
        "links": {
          "self": {
            "href": "xxxxx"
          }
        },
        "code": "produzent_4711",
        "values": {
          "label": [
            {
              "locale": "de_DE",
              "channel": null,
              "data": "Produzent 4711"
            }
          ],
          "address_zip_code": [
            {
              "locale": "de_DE",
              "channel": null,
              "data": "123456"
            }
          ],
          "address_city": [
            {
              "locale": "de_DE",
              "channel": null,
              "data": "Stadt"
            }
          ],
          "address_country": [
            {
              "locale": null,
              "channel": null,
              "data": "Land"
            }
          ]
        }
      }
    ]
  }
}

```

parsingCode für JSON Reader:


```

<#assign row = target.addRow()>
<#list json["_embedded"]["items"] as r >
  <#assign itemsRow = target.addRow()>
  ${itemsRow.addCol("code",r["code"]!)}
  ${itemsRow.addCol("label",r["values"]["label"][0]["data"]!)}
  ${itemsRow.addCol("address_zip_code",r["values"]["address_zip_code"][0]["data"]!)}
  ${itemsRow.addCol("address_city",r["values"]["address_city"][0]["data"]!)}
  ${itemsRow.addCol("address_country",r["values"]["address_country"][0]["data"]!)}
</#list>

```

Ergebnis:

Vorschau des Steps ✕

JSONReader

Vorschau erfolgreich ausgeführt

Die Ausgabe dient zu Vorschauzwecken und kann von der Ausgabe der echten Ausführung abweichen.

SPREADSHEET output@JSON2Spreadsheet_3

Vorschau Filter + Filter anwenden Zeige max. 10 Zeilen

[Gekürzte Spalten erweitern](#)

code	label	address_zip_code	address_city	address_country
produzent_4711	Produzent 4711	123456	Stadt	Land

Beispiel 6 - JSON Parsing

Quelle: <https://forum.synestry.com/t/plenty-newsletter-ordner-abrufen/2387>

Anforderung:

- KundenID in erster Spalte
- dann die enthaltenen Felder (bzw: email und confirmedTimestamp) als weitere Spalten

JSON:

```

{
  "12345": {
    "id": 140937,
    "folderId": "18",
    "contactId": "0",
    "firstName": "redacted",
    "lastName": "redacted",
    "email": "redacted",
    "gender": null,
    "birthday": "0000-00-00",
    "timestamp": "2021-08-07 17:43:27",
    "templateLang": "de",
    "confirmedTimestamp": "2021-08-07 17:44:09",
    "confirmationURL": null
  },
  "67890": {
    "id": 140938,
    "folderId": "18",
    "contactId": "0",
    "firstName": "redacted",
    "lastName": "redacted",
    "email": "redacted",
    "gender": null,
    "birthday": "0000-00-00",
    "timestamp": "2021-08-09 20:24:18",
    "templateLang": "de",
    "confirmedTimestamp": "0000-00-00 00:00:00",
    "confirmationURL": null
  }
}

```

parsingCode für JSON Reader:

```

<#assign row = target.addRow()>
<#list json as key, value>
  <#assign row = target.addRow()>
  ${row.addCol("customerid", key)}
  ${addColumns(row, value, "data_")}
</#list>

```

Ergebnis:

Vorschau des Steps

×

JSONReader

Vorschau erfolgreich ausgeführt

Die Ausgabe dient zu Vorschauzwecken und kann von der Ausgabe der echten Ausführung abweichen.

SPREADSHEET output@JSON2Spreadsheet_1

Vorschau Filter

Gekürzte Spalten erweitern

customerid	data_id	data_folderId	data_contactId	data_firstName	data_lastName	data_email	data_gender	data_birthday	data_timestamp	data_templateLang	data_confirmedTimestamp	data_confirmationURL
12345	140937	18	0	redacted	redacted	redacted		0000-00-00	2021-08-07 17:43:27	de	2021-08-07 17:44:09	
67890	140938	18	0	redacted	redacted	redacted		0000-00-00	2021-08-09 20:24:18	de	0000-00-00 00:00:00	

Weitere Informationen und Besonderheiten

Die Schwierigkeit war, dass dieses JSON-Objekt keine Liste (Array) ist, sondern eine Map aus Key-Value-Paaren ist.
Mit `<#list json as key, value>` kann man über die einzelnen Key-Value-Paare dieser Map iterieren und auf den Key und den Value zugreifen.
Dies ist die [Freemarker-Schreibweise für Key-Value-Paare einer Map / Hash](#).

Mehr zum Thema XML, JSON und API-Anbindungen

Siehe [API Connector Tools](#)