

Import von Webdaten

Jan-Philipp Kolb

8 Mai 2017

1 Import von JSON Dateien

2 Import von XML Dateien

Import von JSON Dateien

JavaScript Object Notation (JSON)

- Jedes gültige JSON-Dokument soll ein gültiges JavaScript sein
- JSON wird zur Übertragung und zum Speichern von strukturierten Daten eingesetzt
- Insbesondere bei Webanwendungen und mobilen Apps wird es in Verbindung mit JavaScript, Ajax oder WebSockets zum Transfer von Daten zwischen dem Client und dem Server häufig genutzt.

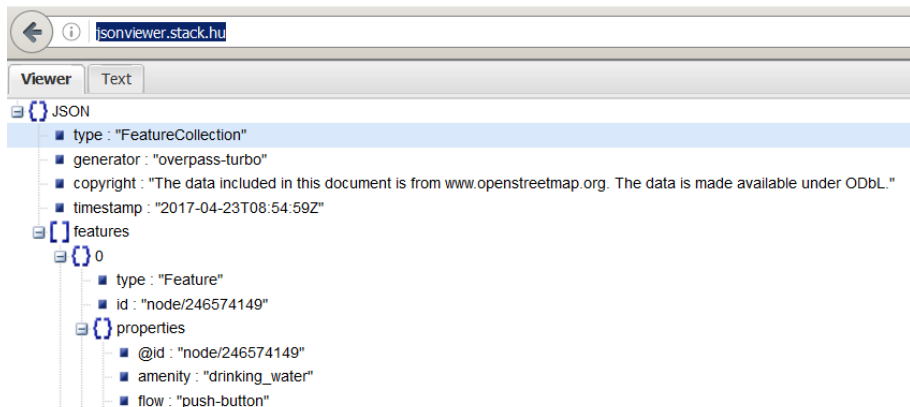
```
{  
  "Herausgeber": "Xema",  
  "Nummer": "1234-5678-9012-3456",  
  "Deckung": 2e+6,  
  "Waehrung": "EURO",  
  "Inhaber":  
    {  
      "Name": "Mustermann",  
      "Vorname": "Max",  

```

Das GeoJSON Format

- GeoJSON ist ein offenes Format um geografische Daten nach der Simple-Feature-Access-Spezifikation zu repräsentieren.
- Dafür wird die JavaScript Object Notation verwendet.

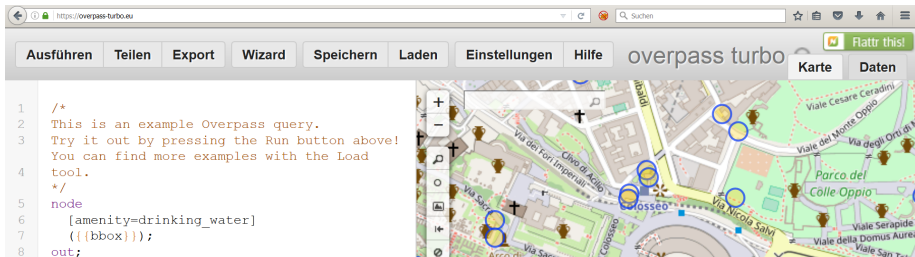
Die Struktur der Daten kann man sich mit einem JSON Viewer anschauen



Download von Beispieldaten

- Overpass Turbo kann verwendet werden um Beispieldaten zu bekommen

<https://overpass-turbo.eu/>











The screenshot shows the Overpass Turbo web interface. The browser address bar displays <https://overpass-turbo.eu/>. The interface includes a navigation bar with buttons: "Ausführen", "Teilen", "Export", "Wizard", "Speichern", "Laden", "Einstellungen", and "Hilfe". The "Ausführen" button is highlighted. To the right of the navigation bar is a search bar with the placeholder text "Suchen". Below the navigation bar, there are two tabs: "Karte" and "Daten". The "Karte" tab is active, showing a map of Rome with several blue circles highlighting specific locations. The "Daten" tab is also visible. On the left side of the interface, there is a text area containing an example Overpass query:

```
1 /*
2 This is an example Overpass query.
3 Try it out by pressing the Run button above!
4 You can find more examples with the Load
5 tool.
6 */
7 node
8   [amenity=drinking_water]
9   ({{bbox}});
10 out;
```





Exkurs OpenStreetMap Daten

- Auf Overpass Turbo können Daten für Map Features exportiert werden
- Eine Liste der erhältlichen Map Features gibt es auf http://wiki.openstreetmap.org/wiki/DE:Map_Features

Schlüssel	Wert	Element	Kommentar	Darstellung	Foto
Verpflegung, Einkehr					
amenity	bar		Bar, Nachtlokal. Es werden hauptsächlich alkoholische Getränke serviert. Siehe auch Beschreibung von amenity=bar und amenity=pub zur Unterscheidung von Bar und Pub (Kneipe).		
amenity	bbq		Grillplatz. Kann mit fuel=* (wood/gas/electric) kombiniert werden.		
amenity	biergarten		Biergarten		

Beispiele für GeoJSON

Einfache Geometrien

Typ	Beispiel	
Point (Punkt)		<pre>{ "type": "Point", "coordinates": [30, 10] }</pre>
LineString (Linie)		<pre>{ "type": "LineString", "coordinates": [[30, 10], [10, 30], [40, 40]] }</pre>
Polygon		<pre>{ "type": "Polygon", "coordinates": [[[30, 10], [40, 40], [20, 40], [10, 20], [30, 10]]] }</pre>
		<pre>{ "type": "Polygon", "coordinates": [[[35, 10], [45, 45], [15, 40], [10, 20], [35, 10], [20, 30], [35, 35], [30, 20], [20, 30]]] }</pre>

JSON importieren

```
library("jsonlite")  
DRINKWATER <- fromJSON("data/RomDrinkingWater.geojson")
```

```
names(DRINKWATER) [1:3]
```

```
## [1] "type"          "generator" "copyright"
```

```
names(DRINKWATER) [4:5]
```

```
## [1] "timestamp" "features"
```

Die Daten anschauen

```
head(DRINKWATER$features)
```

```
##           type           id properties.@id properties.amenity
## 1 Feature node/246574149 node/246574149   drinking_water
## 2 Feature node/246574150 node/246574150   drinking_water
## 3 Feature node/246574151 node/246574151   drinking_water
## 4 Feature node/248743324 node/248743324   drinking_water
## 5 Feature node/251773348 node/251773348   drinking_water
## 6 Feature node/251773551 node/251773551   drinking_water
## properties.type properties.name properties.name:fr proper
## 1           nasone           <NA>           <NA>
## 2           <NA>           <NA>           <NA>
## 3           <NA>           <NA>           <NA>
## 4           <NA>           <NA>           <NA>
## 5           nasone           <NA>           <NA>
## 6           <NA>   Acqua Marcia   Eau potable
```

Github JSON Daten

- Es lassen sich auch Dinge aus dem Web auslesen:

```
my_repos <- fromJSON("https://api.github.com/users/japhilko/repos")
names(my_repos)
```

```
## [1] "id" "name" "full_name"
## [4] "owner" "private" "html_url"
## [7] "description" "fork" "url"
## [10] "forks_url" "keys_url" "collaborators_url"
## [13] "teams_url" "hooks_url" "issue_events_url"
## [16] "events_url" "assignees_url" "branches_url"
## [19] "tags_url" "blobs_url" "git_tags_url"
## [22] "git_refs_url" "trees_url" "statuses_url"
## [25] "languages_url" "stargazers_url" "contributors_url"
## [28] "subscribers_url" "subscription_url" "commits_url"
```

Weiteres Beispiel für JSON Daten

- Die Ergast Developer API ist ein experimenteller Web Service, der eine historische Aufzeichnung von Motorsportdaten liefert.



Ergast Daten lesen

```
library(jsonlite)
res <- fromJSON('http://ergast.com/api/f1/2004/1/results.json')
drivers <- res$MRData$RaceTable$Races$Results[[1]]$Driver
colnames(drivers)
```

```
## [1] "driverId"          "code"              "url"              '
## [5] "familyName"        "dateOfBirth"      "nationality"      '

```

Daten der New York Times

- Die New York Times hat mehrere APIs als Teil des NYT-Entwickler-Netzwerks.
- Es ist eine Schnittstelle zu Daten aus verschiedenen Abteilungen, wie Nachrichtenartikel, Buchbesprechungen, Immobilien, etc.
- Registrierung ist erforderlich (aber kostenlos) und ein Schlüssel kann hier erhalten werden.

New York Times Beispiel

```
article_key <- "&api-key=c2fede7bd9aea57c898f538e5ec0a1ee:6:68"
url <- "http://api.nytimes.com/svc/search/v2/articlesearch.js"
req <- fromJSON(paste0(url, article_key))
articles <- req$response$docs
colnames(articles)
```

```
## [1] "web_url" "snippet" "lead_paragraph"
## [4] "abstract" "print_page" "blog"
## [7] "source" "multimedia" "headline"
## [10] "keywords" "pub_date" "document_type"
## [13] "news_desk" "section_name" "subsection_name"
## [16] "byline" "type_of_material" "_id"
## [19] "word_count" "slideshow_credits"
```

Import von XML Dateien

Das XML Paket

```
library(XML)
citation("XML")
```

```
##
## To cite package 'XML' in publications use:
##
##   Duncan Temple Lang and the CRAN Team (2016). XML: Tools for
##   Parsing and Generating XML Within R and S-Plus. R package
##   version 3.98-1.5. https://CRAN.R-project.org/package=XML
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {XML: Tools for Parsing and Generating XML Within R and S-Plus},
##     author = {Duncan Temple Lang and the CRAN Team},
##     year = {2016}.
```

Erstes Beispiel

```
url <- "http://api.openstreetmap.org/api/0.6/
relation/62422"
```

```
library(xml2)
BE <- xmlParse(url)
```

```
-<osm version="0.6" generator="CGIMap 0.4.0 (19884 thorn-03.openstreetmap.org)" copyright="OpenStreetMap and contributors" attribution="http://www.openstreetmap.org/copyright"
license="http://opendatacommons.org/licenses/odbl/1-0/">
- <relation id="62422" visible="true" version="209" changeset="36072269" timestamp="2015-12-20T19:49:52Z" user="tbicr" uid="278800">
  <member type="node" ref="240109189" role="admin_centre"/>
  <member type="way" ref="50291800" role="outer"/>
  <member type="way" ref="77913336" role="outer"/>
  <member type="way" ref="315222039" role="outer"/>
  <member type="way" ref="77487568" role="outer"/>
  <member type="way" ref="315222038" role="outer"/>
  <member type="way" ref="98035898" role="outer"/>
  <member type="way" ref="77501737" role="outer"/>
```

Das XML analysieren

- Tobi Bosede - Working with XML Data in R

```
xmltop = xmlRoot(BE)  
class(xmltop)
```

```
## [1] "XMLInternalElementNode" "XMLInternalNode"  
## [3] "XMLAbstractNode"
```

```
xmlSize(xmltop)
```

```
## [1] 1
```

```
xmlSize(xmltop[[1]])
```

```
## [1] 326
```

Nutzung von Xpath

Xpath, the XML Path Language, is a query language for selecting nodes from an XML document.

```
xpathApply(BE,"//tag[@k = 'source:population']")
```

```
## [[1]]
```

```
## <tag k="source:population" v="http://www.statistik-berlin-b  
##
```

```
##
```

```
## attr(,"class")
```

```
## [1] "XMLNodeSet"
```

Node parsen

```
url2 <- "http://api.openstreetmap.org/api/0.6/node/2923760808"  
RennesBa <- xmlParse(url2)
```

Way parsen

```
url3 <- "http://api.openstreetmap.org/api/0.6/way/72799743"  
MadCalle <- xmlParse(url3)
```

Mehr Beispiele, wie man mit XML Daten umgeht:

- Daten aus XML extrahieren

<http://www.stat.berkeley.edu/~statcur/Workshop2/Presentations/XML.pdf>

- Duncan Temple Lang - A Short Introduction to the XML package for R

<http://www.omegahat.net/RXML/shortIntro.pdf>

Noch mehr Informationen

- Web Daten manipulieren

<http://www.di.fc.ul.pt/~jpn/r/web/index.html#parsing-xml>

- Tutorial zu xquery

http://www.w3schools.com/xml/xquery_intro.asp

- R und das Web (für Anfänger), Teil II: XML und R

<http://giventhedata.blogspot.de/2012/06/r-and-web-for-beginners-part-ii-xml-in.html>

- String Manipulation

http://gastonsanchez.com/Handling_and_Processing_Strings_in_R.pdf

Referenzen

```
citation("XML")
```

```
##  
## To cite package 'XML' in publications use:  
##  
## Duncan Temple Lang and the CRAN Team (2016). XML: Tools for  
## Parsing and Generating XML Within R and S-Plus. R package  
## version 3.98-1.5. https://CRAN.R-project.org/package=XML  
##  
## A BibTeX entry for LaTeX users is  
##  
## @Manual{,  
##   title = {XML: Tools for Parsing and Generating XML Within  
##   author = {Duncan Temple Lang and the CRAN Team},  
##   year = {2016},  
##   note = {R package version 3.98-1.5}.
```

Links

- XML parsen - Stackoverflow
- Processing of GeoJson data in R