ceu-economics-and-business.github.io

Different Shapes of Data. APIs. – ECBS 5146 SQL and Different Shapes of Data

9–12 minutes

Overview

Teaching: 90 min

Questions

- What the popular flat file formats used by data analysts?
- How I get data from the internet?

Objectives

- Familiarize popular data file format
- Build data files in different formats
- Understanding URLs used on the internet
- Introducing APIs as data sources
- Exercise REST APIs

Keywords

#CSV

#XML

#JSON

#API

#POSTMAN

#EUROSTAT API

#WORLDBANK API

Table of Content

Prerequisites for this chapter

- Install: Postman
- Read: Chapter: “When Databases Attack: A Guide for When to Stick to Files” from “Bad Data Handbook”

Flat files as data store

Advantages

- No additional software required other than OS built in

- You can start exploring the data right away
- Store in any format, invent your own
- Simple
- (Most of them are) Human readable

Disadvantages

- Security
- Self managed (e.g. backup)
- Encoding
- Learning time
- Scalability

*.csv

First Name	Last Name	Job Title
Kate	Middleton	Princess
Bill	Gates	Retired
Ronald	McDonald	Clown

Format

```
Column name1,Column name2 --> <Header_line>
Value1,Value2 --> <Record_1_line>
Value3,Value4 --> <Record_2_line>
```

- CSV stands for **comma-separated values**
- Used to store tabular data
- Not the *other Excel format* (Popular spreadsheet tools can handle it, but it is not recommended for serious analysts)

- Not standardized, but has some common characteristics:
- Optional header
- Each line after header is a record
- Record has as much values as defined in the header
- Values are separated by comma, yet other separators are also allowed (often semicolons)
- Values might be quoted, to avoid confusion with separator
- Beware: line ending can vary from OS to OS, so during the processing you might need to try multiple combinations (\n\r)

Example

```
First Name,Last Name,Job Title
Kate,Middleton,Princess
Bill,Gates,Retired
Ronald,McDonald,Clown
```

*.xml

Format

```
<element>information</element>
<element attribute=“information”>information</
element>
<element attribute=“information”/>
<element>
        <nested-element>information</nested-element>
</element>
```

- XML stands for **Extensible Markup Language**

- Used to exchange tabular and hierarchical data on web
- The operation of loading in code is called parsing
- Webpage markup XHTML is a subset of XML
- Criticism: verbose, complex and redundant
- Alternative: JSON

Example - Variation 1

```
<persons>
  <person>
            <first_name>Kate</first_name>
            <last_name>Middleton</last_name>
            <job_title>Princess</job_title>
  </person>
  <person>
            <first_name>Bill</first_name>
            <last_name>Gates</last_name>
            <job_title>Retired</job_title>
  </person>
  <person>
            <first_name>Ronald</first_name>
            <last_name>McDonald</last_name>
            <job_title>Clown</job_title>
  </person>
</persons>
```

Example - Variation 2

```
<persons>
```

```
  <person>
        <name first="Kate" last="Middleton"/>
        <job title="Princess"/>
  </person>
  <person>
        <name first="Bill" last="Gates"/>
        <job title="Retired"/>
  </person>
  <person>
        <name first="Ronald" last="McDonald"/>
        <job title="Clown"/>
  </person>
</persons>
```

Example - Variation 3

```
<persons>
  <person first_name="Kate" last_name="Middleton"
job_title="Princess"/>
  <person first_name="Bill" last_name="Gates"
job_title="Retired"/>
  <person first_name="Ronald" last_name="McDonald"
job_title="Clown"/>
</persons>
```

*.json

- JSON is an acronym for **JavaScript Object Notation**, is an open standard format, which is lightweight and text-based, designed explicitly for human-readable data interchange.

- Despite of having *JavaScript* in the name, JSON is a coding language independent format
- JSON is the most widespread standard for exchanging data on the web, therefore, mastering it is key for data analysts.

Example - The previous Variation 3 in JSON

```
{
   "persons": [
       {
           "first name": "Kate",
           "last name": "Middleton",
           "job title": "Princess"
       },
       {
           "first name": "Bill",
           "last name": "Gates",
           "job title": "Retired"
       },
       {
           "first name": "Roland",
           "last name": "McDonalds",
           "job title": "Clown"
       }
   ]
}
```

JSON validation

Learning JSON takes time and practice. The learning curve is

higher than in case of CSV or XML.

In this learning process a validation tool could be really handy: https://jsonformatter.curiousconcept.com/#. Use this to validate the JSON structures you are creating during the exercises.

Elements of JSON format: Key/Value data

In the heart of JSON is the key value data represented like this:

key - is represented by a string enclosed in single or double quotation marks.

value - is typically represented a data type.

Elements of JSON format: Data types

A value can use the following data types:

Strings: Characters that are enclosed in single or double quotation marks. Example:

Number: A number could be integer or decimal, positive or negative. Example:

Booleans: The Boolean value could be either true or false without any quotation marks. Example:

Null: Here, null means nothing without any quotation marks. Example:

Elements of JSON format: The JSON Object

Objects are JSON elements that are enclosed in curly brackets. In objects, keys and values are separated by a colon ':', pairs are separated by comma. This is the smallest JSON possible (one

JSON Object with one key/value):

Now, here is a JSON Object with two key/value:

```
{"key1":"value1","key2":"value2"}
```

Besides the data types above, a value can be a JSON Object as well (nesting):

```
{"key1":{"key2":"value2"}}
```

Elements of JSON format: The JSON Arrays

To complicate further, besides the data types and JSON Objects a value can be an array as well. Arrays are the lists that are represented by the square brackets, and the values have commas in between them. The value of an array can be the well known data types or JSON objects or another array. We can also mix these i.e., a single array can have strings, boolean, numbers or objects:

```
{"example1":[1, 2, 7.8, 5, 9, 10]}
{"example2":["red", "yellow", "green"]}
{"example3":[{"key1":30}, {"key2":40}]}
{"example4":[1,2,[56,57]]}
{"example5":[8, "hello", null, true, {"key1":30}]}
```

Elements of JSON format: The JSON Document

The root element (top of the tree hierarchy) in a .json file should be always one JSON object or a JSON Array. This said the following JSON document is invalid:

```
{"key1":"value1"},{"key2":"value2"}
```

but this one is valid:

```
[{"key1":"value1"},{"key2":"value2"}]
```

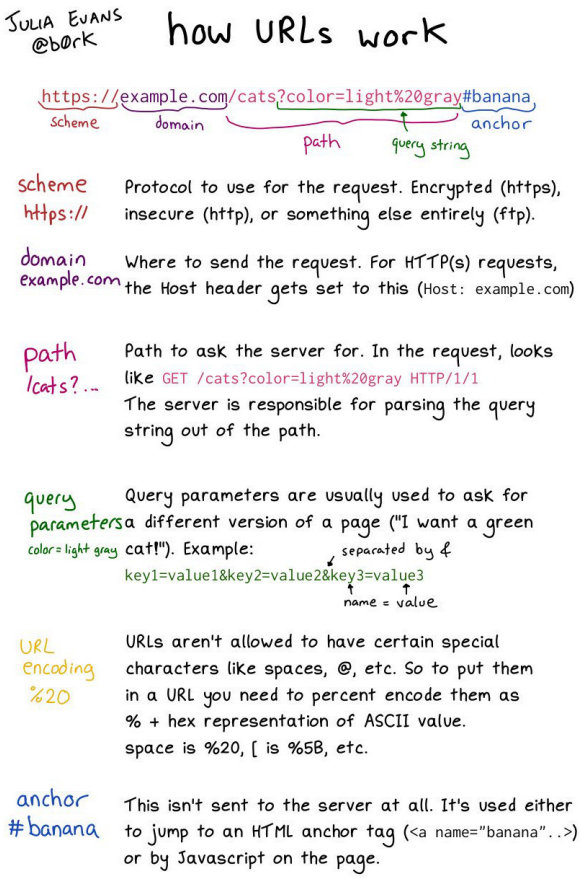
Exercise 1 XML to JSON exercise

Convert bookstore.xml to JSON.

Exercise 2 JSON design exercise

Create a JSON file which represents best the information we stored for “University” DB (Students, Programs, Courses etc)

Anatomy of a URL



Getting data from internet

Web scraping

- Data scraping used for extracting data from websites
- Its is not the most robust way to get data (website might change, legal issues)
- Anyway sometimes this is the only way to get the data
- Beyond the scope of this course, you will have lessons later in the program

Data through APIs

- Right way to offer data
- Instead of a webpage, server returns a flat file in various formats (csv, json etc)
- Simplest way is REST
- Sometimes is password protected

> ## Example - Eurostat API
>
> Eurostat is offering a wide range of databases: https://ec.europa.eu/eurostat/web/main/data/database
>
> - Navigate to: Data navigation tree > Detailed datasets > General and regional statistics > City statistics (urb) > Cities and greater cities (urb_cgc) > Population on 1 January by age groups and sex - cities and greater cities (urb_cpop1)
> - Click on Data Browser (1st icon) and you'll get this: https://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=urb_cpop1&lang=en

- Check the dimensions of this dataset, play around with slicing and dicing
- Browse to https://wikis.ec.europa.eu/display/EUROSTATHELP/API+-+Getting+started+with+statistics+API and read it carefully.
- Now lets build our first query with this: https://ec.europa.eu/eurostat/web/query-builder/getting-started/query-builder
- Insert the db name: “urb_cpop1” and click next
- Now you can specify some dimension values like: Geo selection > Fixed > Bruxelles AND DE1003V - Population on the 1st of January, female
- Click “Generate query filter” and you’ll get this URL: https://ec.europa.eu/eurostat/api/dissemination/statistics/1.0/data/urb_cpop1?format=JSON&indic_ur=DE1003V&cities=BE001C&lang=en
- Try this URL in a browser. What do you see?
- The more convenient way to work with REST APIs is Postman. Lets paste the previous URL in postman.

`Exercise 3` - Worldbank API

Using World Bank Data API, request “GDP per capita, PPP (current international $)” [indicator:NY.GDP.PCAP.PP.CD] for all 266 countries and regions.

Hints:

- The information requested is from “World Development Indicators” dataset, which can be browsed here
- More details in the API you can find here

`Exercise 4` - Worldbank API Advanced

Using World Bank Data API, request “CO2 emissions (metric tons per capita)” between 2000 and 2002 for EU,Hungary, USA and China.