OPEN DATA E DBMS
FORMATO JSON E XML

SISTEMI INFORMATIVI E DBMS

CORSO DI LAUREA MAGISTRALE IN
MANAGEMENT E MONITORAGGIO DEL TURISMO SOSTENIBILE

PROF. ANDREA PINNA

AA 2019/2020

JAVASCRIPT OBJECT NOTATION

Il formato Json è un formato testuale che riproduce la sintassi del linguaggio javascript per la definizione di dati. Lo standard attuale è il RFC8259

https://tools.ietf.org/html/rfc8259

JSON: STRUTTURA DATI

I dati serializzati nel formato JSON sono strutturati in due modi, che possono essere tra loro combinati:

Array: `[ dato1, dato2 ]`
Object: `{chiave1:valore1, chiave2:valore2}`

Gli elementi dell'array vengono chiamati membri.
Come si vede, l'array equivale alla lista python, l'object equivale al dizionario python.

JSON: STRUTTURA DATI

Esempio estratto da `anagrafica_biblioteche.json`

:

```
[{"DENOMINAZIONE":"Arbus - Biblioteca comunale",
"CAP":"09031"},{"DENOMINAZIONE":"Armungia -
Biblioteca comunale Emilio
Lussu","CAP":"09040"}]
```

Secondo lo standard JSON questa riga del file rappresenta un array di object.

JSON: STRUTTURA DATI

La struttura dei dati è tipicamente annidata. Spesso viene presentata con specifiche indentazioni che rendono più agevole la lettura. Esempio:

```
[
    {
        "DENOMINAZIONE":"Arbus - Biblioteca comunale",
        "CAP": "09031"
    },
    {
        "DENOMINAZIONE":"Armungia - Biblioteca comunale
Emilio Lussu",
        "CAP":"09040"
    }
]
```

JSON CON PYTHON

Il formato JSON è supportato dal linguaggio Python tramite il modulo `json`.

Il modulo permette di serializzare un qualunque dato, trasformandolo in una stringa in formato JSON. Permette inoltre di deserializzare i dati presenti in una stringa in formato JSON per usarli nel programma.

Documentazione del modulo:
https://docs.python.org/3.7/library/json.html

JSON CON PYTHON

Per usare il modulo json dobbiamo importarlo.

```
import json
```

Vedremo le seguenti funzioni del modulo:

```
json.dumps
json.loads
```

JSON CON PYTHON

La conversione di tipo da JSON a Python e viceversa è quasi univoca.
Fanno eccezione i numeri che su JSON hanno un solo tipo, mentre su Python si distinguono in float e int

JSON	Python
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

JSON.DUMPS

La funzione `json.dumps` è la funzione di serializzazione che prende in ingresso un qualsiasi dato del formato Python (ad sempio un dizionario, una lista, o un dato annidato) e restituisce una stringa in formato JSON.

Esempio:

```
import json
stringaJson=json.dumps([{"id":2,"Elenco":[3,5,7]}])
```

Ora `stringaJson` contiene una stringa con i dati serializzati nel formato JSON

JSON.DUMPS CON INDENTAZIONE

Con la funzione `json.dump` è anche possibile creare la stringa formato JSON con un una formattazione più comprensibile. In particolare si più far andare a capo e indentare i vari dati.
Per fare questo è sufficiente inserire un secondo argomento alla funzione json.dumps assegnando un valore al parametro `indent`.

Esempio:

```
stringaJson=json.dumps(dato,indent=4)
```

JSON.DUMPS CON INDENTAZIONE

Esempio dalla console:

```
>>> import json
>>> a=json.dumps([{"id":7}],indent=4)
>>> print(a)

[
    {
        "id": 7
    }
]
```

JSON.LOADS

La funzione `json.loads` è la funzione di deserializzazione dei dati dal formato JSON. Prende in ingresso una stringa in formato JSON e restituisce i dati "tradotti" nei tipi di dato Python.

```
dati = json.loads('[{"DENOMINAZIONE":"Arbus - Biblioteca comunale", "CAP":"09031"}]')
```

In questo semplice esempio, la variabile `dati` conterrà una lista di un solo elemento. Questo elemento è un dizionario con due coppie chiave-valore

JSON.LOADS

Esempio dalla console:

```
>>> import json
>>> dati = json.loads('[{"DENOMINAZIONE":"Arbus
-  Biblioteca comunale", "CAP":"09031"}]')

#Ora posso accedere al primo elemento della lista e
accedo all'elemento del dizionario con chiave "CAP"

>>> print(dati[0]["CAP"])
09031
```

JSON CON PYTHON: I FILE

Abbiamo visto che il modulo json elabora stringhe. Ma sappiamo che i dati json possono essere registrati dentro i file di testo.

Per scrivere i dati in formato JSON su un file è sufficiente scrivere la stringa ottenuta con `json.dumps` all'interno di un file usando il metodo write del wrapper del file aperto in modalità "w".

JSON CON PYTHON: I FILE

Esempio: salviamo su un file chiamato "`dati.json`" un dato python serializzato in formato JSON.

```
import json
mioDato = [{"id":2,"Elenco":[True,5,None]}]
stringaJson = json.dumps(mioDato,indent=4)
with open("dati.json","w") as fileDaScrivere:
    fileDaScrivere.write(stringaJson)
```

JSON CON PYTHON: I FILE

Per leggere i dati serializzati in un file JSON dobbiamo aprire il file in modalità lettura e inserire in un unica stringa, tutto il contenuto del file. Possiamo poi tradurre la stringa JSON in dati Python usando la funzione `json.loads` vista prima. Esempio:

```
import json
stringaJson=""
with open("dati.json","r") as fileDaLeggere:
    for line in fileDaLeggere:
        stringaJson+=line
datoEstratto=json.loads(stringaJson)
```

JSON: ESERCIZIO

Sacaricare il file `"anagrafica_biblioteche.json"` ed elaborarlo con Python per determinare:

- Il numero di biblioteche in provincia di Nuoro,
- l'elenco delle tipologie funzionali ed il numero di biblioteca per ogni tipologia.

- Infine convertire tutti i dati del file JSON in un file formato CSV.

DATA PARZIALE

Collegatevi e votate la data.

https://doodle.com/poll/b7qh85cthqn7sgim

➜ La decisione finale è vincolata dalla disponibilità delle aule. ←

Se non si potesse cambiare data o se non si trovasse un accordo, rimane valida la data del 14 novembre.

EXTENSIBLE MARKUP LANGUAGE

Il formato XML nasce come linguaggio estensibile. Gli elementi del linguaggio sono chiamati tag e possono essere definiti a piacimento.

Un tag è un'etichetta che serve a dare significato ad un dato (che può essere testo od altro),

Documentazione di riferimento:
https://www.w3.org/TR/REC-xml/

XML: I TAG

I tag si creano e si aprono scrivendo un nome inserito tra "parentesi angolari" ovvero tra i caratteri `< e >`.
Esempio: creo e apro un tag.

```
<valoreBase>
```

I tag si chiudono inserendo lo stesso nome tra i simboli `</ e >.` Esempio: chiudo il tag.

```
</valoreBase>
```

XML: I TAG

Il valore del tag si deve inserire tra l'apertura e la chiusura del tag.

Esempio: Voglio dare il valore "10" al tag.

```
<valoreBase>10</valoreBase>
```

La riga qui sopra è un documento XML valido.

XML: I TAG

Ogni tag può contenere altri tag.
Esempio:

```
<modelloBase>
   <colore>blu</colore>
   <cilindrata>1297</cilindrata>
</modelloBase>
```

XML: I TAG

Il nome del tag deve rispettare alcune regole:

- Non può iniziare con un numero
- Non può contenere spazi
- Non può iniziare con caratteri speciali.

Inoltre i tag aperti devono essere chiusi e la chiusura non può essere scritta tra l'apertura e la chiusura di un altro tag.

XML: PREAMBOLO E COMMENTI

La prima riga del file XML può contenere il preambolo che contiene informazioni sulla versione e sulla codifica del carattere.
È contenuto tra i simboli `<? e ?>` .

```
<?xml version="1.0" encoding="UTF-8"?>
```

Inoltre si possono inserire commenti. I commenti devono stare tra i simboli `<!-- e -->`

```
<!-- Commento nel file -->
```

XML ESEMPIO

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <row>
    <Fondazione_Ente>Fondazione Andrea
Parodi</Fondazione_Ente>
    <Sede_ente>Cagliari</Sede_ente>
  </row>
  <row>
    <Fondazione_Ente>Fondazione Costantino
Nivola</Fondazione_Ente>
    <Sede_ente>Orani</Sede_ente>
   </row>
</root>
```

XML SU PYTHON

Python fornisce il modulo `xml.dom.minidom` che permette di deserializzare i dati contenuti in un documento xml

I dati vengono estratti e resi compatibili con l'interfaccia Document Object Model (DOM).

Consente inoltre di serializzare i dati DOM in una stringa XML
Documentazione:
https://docs.python.org/3/library/xml.dom.minidom.html

XML SU PYTHON

Per usare il modulo `xml.dom.minidom` dobbiamo prima importarlo.

```
import xml.dom.minidom
```

In questo corso vederemo soltanto come convertire una stringa XML (ad esempio estratta da un file scritto in formato xml) in un dato DOM, Useremo il metodo:

```
.parseString
```

XML PARSESTRING

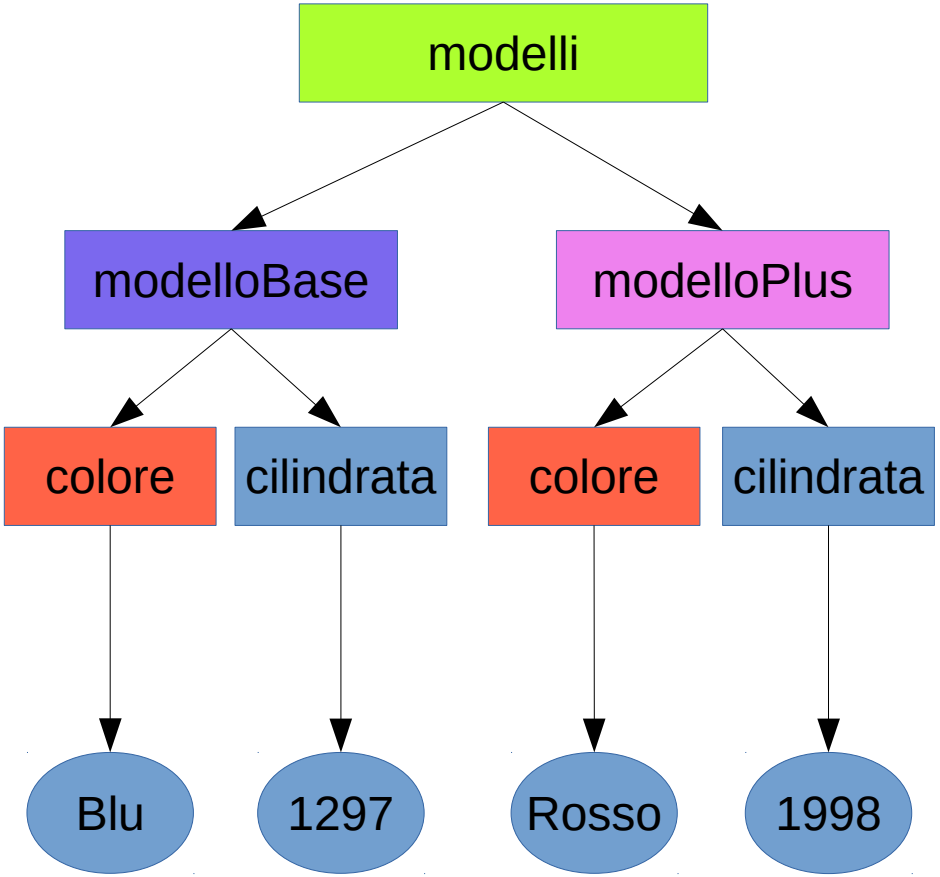
Il metodo `parseString` prende in ingresso una stringa scritta nel formato XML e restituisce un DOM.

Esempio: supponiamo che dentro `document` ci sia una stringa formato XML. Per deserializzarla posso usare:

```
import xml.dom.minidom
mioDom = xml.dom.minidom.parseString(document)
```

XML ESEMPIO DOM

```
document = """
<modelli>
<modelloBase>
<colore>Blu</colore>
<cilindrata>1297</cilindrata>
</modelloBase>
<modelloPlus>
<colore>Rosso</colore>
<cilindrata>1998</cilindrata>
</modelloPlus>
</modelli>
"""
```



XML GET ELEMENTS

Ottenuto l'oggetto DOM, posso usare il metodo `getElementsByTagName` che prende in ingresso una stringa (il nome di un tag) e restituisce una lista di elementi del DOM che corrispondo a quel nome.

Ad esempio, considerando la stringa della slide precedente, il metodo applicato al nome `modelli` restituisce un solo elemento. Se si applica al nome `colore` ottengo una lista di due elementi.

XML GET ELEMENTS

Ciascun elemento ottenuto con il metodo `getElementsByTagName` può contenere i dati.

I contenitori dei dato sono chiamati "child".

Dato un elemento, accedo al suo dato scrivendo:
`nomeElemento.firstChild.data`

XML GET ELEMENTS

Esempio riferito alla stringa `document` vista prima.

```
import xml.dom.minidom

mioDom = xml.dom.minidom.parseString(document)
colori = mioDom.getElementsByTagName("colore")
colore = colori[0] #primo elemento di due
print(colore.firstChild.data)
```

Il programma stamperà la stringa "Blu"

XML GET ELEMENTS

Il metodo getElementsByTagName restituisce una lista di "elements" DOM per un dato tag.
Il nome del tag è un attributo dell'elemento.

```
nomeTag = colori[0].tagName
```

Ogni elemento della lista contiene una lista di childNodes.

```
listaChild = colori[0].childNodes
```

Un childNode può avere un valore:

```
valoreEstratto = listachild[0].firstChild.data
```

XML E FILE

Anche in questo caso, per deserializzare un file XML dobbiamo prima importare il contenuto del file nel programma sotto forma di stringa. Per farlo apriamo il file in modalità lettura e carichiamo il testo in una variabile stringa del programma.

```
document=""
with open("beniCultura.xml", "r", encoding="utf8")
as fileAperto:
   for line in fileAperto:
      document+=line
```

XML ESERCIZIO

Scaricare il file `tourist-info-centers_v2.xml` contenente informazioni sui servizi per il turismo delle Fiandre e Bruxelles.

- Stampare a video il numero di centri (tag element) registrati nel file.
- Stampare a video il nome (tag name) di tutti i centri.
- Stampare a video l'indirizzo (via, numero civico, codice postale, città) e numero di telefono del centro chiamato "Toerisme Beernem".
- Stampare a video indirizzo e numero di telefono di un centro dato il nome richiesto dall'utente come input.

Scrivere un CSV che per ogni "element" riporti i valori dei tag:
`discriminator, name, city_name, phone1, website.`
L'intestazione del file CSV deve essere in italiano.