OPEN DATA E DBMS

DATI SERIALIZZATI
FORMATO CSV

SISTEMI INFORMATIVI E DBMS

CORSO DI LAUREA MAGISTRALE IN
MANAGEMENT E MONITORAGGIO DEL TURISMO SOSTENIBILE

PROF. ANDREA PINNA

AA 2019/2020

DATI SERIALIZZATI

La serializzazione dei dati è un processo che permette di tradurre i dati di un programma in un formato che può essere archiviato in un supporto di memoria o trasmesso in una rete di calcolatori.

Il processo inverso è chiamato deserializzazione e permette di ottenere il dato originale, utilizzabile dal programma.

DATI SERIALIZZATI

I dati rappresentati in un formato serializzato possono essere memorizzata su un file. Questo permette la separazione dei dati dal programma e consente di avere i dati disponibili e trasportabili.

Tutti tipi di dato visti fino ad ora possono essere serializzati e deserializzati, compresi quelli strutturati.

DATI SERIALIZZATI

Il formato dei dati serializzati può essere leggibile sia dalla macchina che dall'uomo (formato testuale) oppure solo dalla macchina (formato binario). Esistono numerosi formati di dati. Nel seguito vedremo tre formati testuali, tutti molto diffusi in ambito professionale. Questi sono:

- CSV
- JSON
- XML

Per un elenco dei formati:
https://en.wikipedia.org/wiki/Comparison_of_data-serialization_formats

COMMA SEPARATED VALUES

Un file testuale formato CSV contiene valori separati da virgole (in inglese, *comma*).

I dati sono organizzati per linee (o righe) e ogni linea contiene un numero indeterminato di valori. La prima riga può essere dedicata all'intestazione e può essere generata in automatico nel processo di serializzazione.

Il formato è anche noto come standard RFC7111 (preceduto da RFC4180

COMMA SEPARATED VALUES

Il formato permette di registrare numeri e stringhe. I numeri possono essere interi o reali. Le stringhe sono delimitate da doppi apici.

Le specifiche del formato non sono rigide. Ad esempio al posto del separatore virgola (`,`) in alcuni sistemi si utilizza il separatore punto e virgola (`;`) come ad esempio nei sistemi microsoft office.

Riferimento:
https://tools.ietf.org/html/rfc7111

COMMA SEPARATED VALUES

Esempio estratto dal file `movimenti_comuni_2017.csv`:

```
"anno","provincia","comune","mese","provenienza","arrivi","presenze"
2017,"OT","TEMPIO PAUSANIA",4,"Regno Unito",7,18
2017,"SS","ALGHERO",11,"Basilicata",2,2
2017,"CI","IGLESIAS",1,"Emilia Romagna",15,56
2017,"CA","ASSEMINI",10,"Francia",32,77
```

I dati si presentano in un formato che ricorda le tabelle tipiche dei programmi di foglio elettronico come LibreOffice Calc o Microsoft Excel e sono facilmente importabili in tali programmi.

CSV CON PYTHON

Python permette di importare i dati serializzati in un file CSV.

Il linguaggio ha un modulo apposito chiamato `csv` che fornisce funzioni per elaborare un file aperto nel programma.

La documentazione relativa al modulo è disponibile in questa pagina: https://docs.python.org/3/library/csv.html

CSV CON PYTHON

Per utilizzare le funzioni del modulo csv occorre prima importare il modulo nel nostro programma usando la parola chiave `import`:

```
import csv
```

Le funzioni del modulo che vedremo sono:

```
csv.reader
csv.writer
csv.DictReader
csv.DictWriter
```

CSV.READER

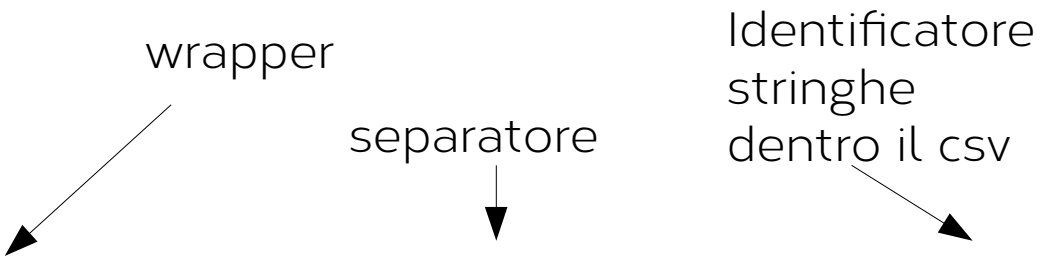
La funzione `csv.reader` ci permette di importare i dati registrati in un file testuale in formato csv.

La funzione prende in ingresso il wrapper del file da leggere, il carattere utilizzato nel file come separatore dei valori (delimiter) e il carattere di identificazione delle stringhe (quotechar) che verrà rimosso dal testo importato.

La funzione csv.reader restituisce un riferimento al contenuto del file che si comporta in modo simile al wrapper del file.

CSV.READER

Esempio d'uso.




```
import csv
fileAperto=open(nomefile,'r')
reader = csv.reader(fileAperto, delimiter=',', quotechar='"')

#Ora posso usare il reader
for row in reader:
    print(row) #stampa una lista di elementi già separati
fileAperto.close()

#Per ripasso, confrontare con il for eseguito sul wrapper
fileAperto
```

CSV.READER

Il file deve essere prima di tutto aperto con la funzione open. In particolare, per usare la funzione csv.reader il file deve essere aperto in modalità lettura.

Successivamente posso chiamare la funzione `csv.reader` che prende come argomenti: il wrapper del file aperto, il carattere delimiter (la virgola che separa i valori del csv) e il quotechar

CSV.READER

Il risultato della funzione csv.reader è un tipo di dato specifico che permette di accedere a tutte le righe del csv.

Funziona come il wrapper del file e quindi posso scrivere un ciclo che itera sugli elementi del reader.

Ogni elemento estratto dal reader corrisponde ad una lista di valori del csv corrispondenti ad una riga. Tutti i valori letti sono considerati stringhe di testo.

I dati della riga sono ora utilizzabili dal programma e quindi la funzione `csv.reader` permette di deserializzare i dati.

CSV.READER ESEMPIO

Esempio: voglio inserire in una lista chiamata `dati` tutte le righe del file csv visto prima. Successivamente voglio stampare la sola colonna 3 dei nomi dei comuni.

```
import csv
#dati=[]
with open('movimenti_comuni_2017.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=',', quotechar='"')
    dati = list(reader) # converto il reader in una lista.
    #Equivale a:
    #for row in reader:
    #  dati+=[row] #metto i dati in una lista
for riga in dati:
    print(riga[2]) #Lavoro sui dati estratti
```

CSV.WRITER

La funzione `csv.writer` permette di scrivere nel file csv in maniera semplificata.

Prende in ingresso una wrapper di file (aperto in una delle modalità di scrittura) e il delimitatore desiderato.

Il risultato di csv.writer è un manipolatore del file csv che contiene il metodo `writerow`. Questo metodo trasforma una lista di dati in una stringa da scrivere sul file csv.

La funzione csv.writer trasforma i dati usati dal programma in un file di testo csv. Quindi questa funzione consente la serializzazione dei dati.

CSV.WRITER

Esempio. Apro il file con il costrutto `with`.

```
import csv
with open('mieidati.csv','w') as csvfile:
    writer = csv.writer(csvfile, delimiter=',')
    writer.writerow(['anno','tipo','dettaglio','codice'])
    writer.writerow([2019,'Test','Scrittura',12])
```

Genero una variabile `writer` con la funzione `csv.writer`.
Ora posso scrivere i dati come righe del file csv.

CSV.DICTREADER

La funzione `DictReader` permette estrarre i campi del file csv aperto in modalità lettura. La funzione `DictReader` restituisce un tipo di dato che per ogni riga si comporta in modo simile al dizionario. Posso infatti selezionare un elemento della riga in base al nome e non con l'indice. I nomi dei campi sono scritti nell'intestazione nella prima riga del file e sono separati da virgole.

Esempio:

```
import csv
with open('movimenti_comuni_2017.csv', newline='') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
        print(row['provincia'], row['arrivi'])
```

CSV.DICTWRITER

La classe `DictWriter` restituisce un manipolatore del file che permette di scrivere nel file usando il metodo `writerow` che pernde in ingresso un dizionario ed è utile per specificare in quali campi inserire ciascun dato della riga. Permette inoltre di creare l'intestazione (header) del file csv.

Esempio:

```
import csv
with open('filosofi.csv', 'w', newline='') as csvfile:
    nomeCampi = ['nome', 'cognome']
    writer = csv.DictWriter(csvfile, fieldnames=nomeCampi)
    writer.writeheader()  #scrive l'intestazione
    writer.writerow({'nome': 'Bertrand', 'cognome': 'Russell'})
    writer.writerow({'cognome': 'Sartre','nome': 'Jean-Paul'})
    writer.writerow({'cognome': 'Kant'})
    writer.writerow({'nome': 'Georg Wilhelm Friedrich'})
```

CSV: ESERCIZIO

Dato il file `movimenti_comuni_2017.csv`, vogliamo ottenere per ogni nome di comune la somma delle presenze e degli arrivi.
Per registrare le presenze, creiamo un dizionario che ad ogni nome di comune associa il numero di arrivi e di presenze.

Ora scriviamo un nuovo file `totalePresenze.csv` che ha come intestazione la riga:

```
"Comune, Totale Arrivi, Totale Presenze"
```

Le altre righe del file csv saranno i nomi dei comuni e la somma degli arrivi e delle presenze. Riempiamo il file con gli elementi del dizionario.