Azzolini Riccardo 2020-05-05

# JSON

## 1 Introduzione

Negli ultimi anni, accanto a XML viene utilizzato un altro formato (serializzazione) per dati semi-strutturati: **JSON** (**JavaScript Object Notation** – come suggerisce il nome, è derivato dal formato dei dati del linguaggio JavaScript).

Un documento JSON è tipicamente un **dizionario**, cioè un insieme di coppie chiave-valore dove ogni valore può essere, a sua volta, un documento JSON. Tale dizionario è chiamato **oggetto**.

### 1.1 Struttura dati dizionario

In generale, la struttura dati chiamata *dizionario* è una collezione (solitamente un insieme) di coppie (*chiave*, *valore*), che supporta le seguenti operazioni fondamentali:

- *inserimento*;
- *cancellazione*;
- *ricerca* per chiave, che restituisce il valore corrispondente.

Questa struttura dati ha molti nomi alternativi (array associativo, mappa, symbol table, ecc.), ed è disponibile in quasi tutti i linguaggi di programmazione. Ad esempio:

- in Java, corrisponde all'interfaccia `Map` (la cui implementazione più usata è la classe `HashMap`);
- in Python, è il tipo predefinito `dict`;
- in JavaScript, tutti gli oggetti sono implementati come dizionari (questo è il motivo per cui i dizionari JSON sono chiamati oggetti).

## 2 Proposta di modello dei dati

Attualmente (2020), non è ancora stato definito un modello dei dati standard per i documenti JSON. In seguito, viene presentato (in modo sintetico) uno dei modelli proposti.

Questo modello prevede 7 tipi di valori:

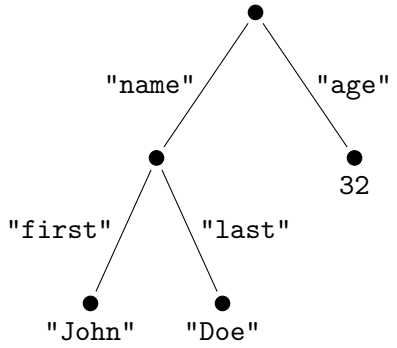
- *stringhe*,
- *numeri*,
- `true`,
- `false`,
- `null`,
- *array* (di altri valori),
- *oggetti* (che, come già detto, sono insiemi di coppie chiave-valore, dove i valori possono essere di tutti questi tipi).

Dato che gli array/oggetti contengono altri valori, i quali possono a loro volta essere array/oggetti, si ha un modello dei dati ad albero "labellato" (etichettato), simile a quelli definiti per XML, ma con alcune importanti differenze (che verranno illustrate più avanti).

Ad esempio, il documento

```
{
    "name": {
        "first": "John",
        "last": "Doe"
    },
    "age": 32
}
```

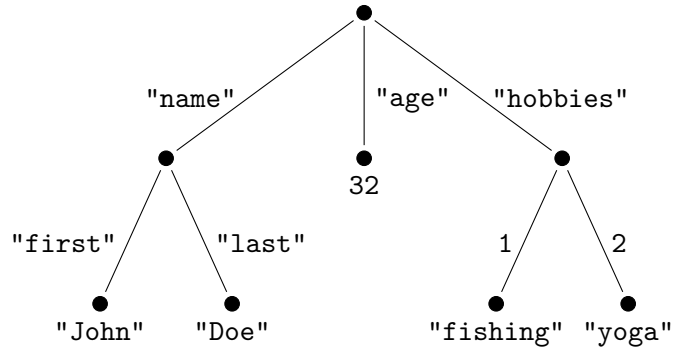
è un oggetto contenente due chiavi (`"name"` e `"age"`); il valore associato alla chiave `"name"` è un altro oggetto, che ha a sua volta due chiavi (`"first"` e `"last"`). L'albero corrispondente, secondo la proposta di modello dei dati presa in considerazione, è:




Per rappresentare gli array, ad esempio il valore associato alla chiave `"hobbies"` nel seguente documento,

```
{
    "name": {
        "first": "John",
        "last": "Doe"
    },
    "age": 32,
    "hobbies": ["fishing", "yoga"]
}
```

si indicano come label gli indici di ciascuno degli elementi dell'array:




## 2.1 Differenze rispetto a XML

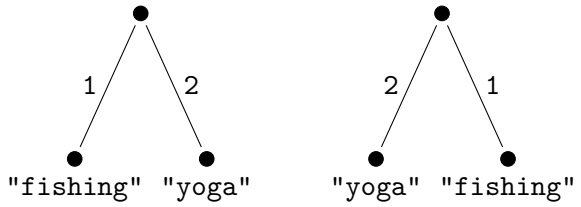
Apparentemente, un documento JSON rappresenta un albero simile a quelli rappresentati dalle serializzazioni XML. Ci sono però alcune importanti differenze:

- Uno stesso oggetto non può avere due chiavi con lo stesso nome. Ad esempio, il seguente documento non è corretto:

```
{
    "name": {
        "first": "John",
        "last": "Doe"
    },
    "age": 32,
    "hobby": "fishing",
    "hobby": "yoga"
}
```

- JSON permette un utilizzo esplicito degli array (in XML, li si può invece simulare usando sequenze di più elementi con lo stesso nome).
- Gli alberi XML sono ordinati, mentre gli alberi JSON no (anche se la serializzazione è intrinsecamente ordinata: in particolare, gli indici degli elementi di un array dipendono dall'ordine di tali elementi nella serializzazione).

  Ad esempio, i seguenti sono considerati identici dal punto di vista del modello dei dati,

  perché l'ordine degli elementi dell'array è definito dalle label.
- Il valore di ciascun nodo dell'albero JSON è l'intero sottoalbero avente tale nodo come radice. Invece, in XML, si considera tipicamente "valore" di un nodo la stringa e/o l'insieme di valori degli attributi associati a esso, senza quindi includere gli eventuali sottoelementi.

## 3 Pro e contro

Un modello dei dati semi-strutturato basato su JSON presenta alcuni vantaggi rispetto a XML:

- La serializzazione è molto più leggera, meno verbosa.
- Il fatto che un oggetto non possa avere due chiavi con lo stesso nome semplifica e velocizza l'attraversamento dell'albero.

Ci sono però anche degli svantaggi:

- Il fatto che il valore di un nodo sia l'intero sottoalbero complica di parecchio le operazioni di confronto tra nodi (ad esempio, per controllare l'uguaglianza).
- Un ultimo svantaggio, anche se questo non dipende dal modello dei dati, è che, al momento, non esiste un linguaggio di interrogazione standard (neanche de-facto) per JSON.

  Nell'ambito della rappresentazione di dati persistenti, JSON è il formato nativo scelto da quasi tutti i *DBMS NoSQL basati su documenti* (tra i quali il più famoso è MongoDB), ma ciascuno di essi adotta un linguaggio di interrogazione diverso.