XML & Json

Java Foundation

Presented by

Valerio Cammarota

IBM Client Innovation Center - Italy

15/11/2023

IBM Client Innovation Center
Italy

Agenda

1. Rappresentazione dei dati
2. XML
3. JSON
4. Differenze tra XML e JSON

IBM Client Innovation Center
Italy

Rappresentazione dei dati

- Sono formati di **dati strutturati**;
- Il loro impiego consente lo **scambio di dati e messaggi** tra piattaforme diverse.

```xml
<note>
  <date> 15-11-2023 </date>
  <hour>09:00</hour>
  <to> Studends </to>
  <from>IBM</from>
  <body>This is XML!</body>
</note>
```

```json
{
   "note": {
          "date": "15-11-2023",
          "hour": "09:00",
          "to": "IBM Academy Studends",
          "from": "IBM",
          "body": "This is JSON!"
     }
}
```

XML

- **XML** sta per e**X**tensible **M**arkup **L**anguage;
- **XML** è un linguaggio di markup;
- **XML** è stato progettato per archiviare e trasportare dati;
- **XML** è stato progettato per essere auto-descrittivo.

```
<note>
  <date> 15-11-2023 </date>
  <hour>09:00</hour>
  <to> Studends </to>
  <from>IBM</from>
  <body>This is XML!</body>
</note>
```

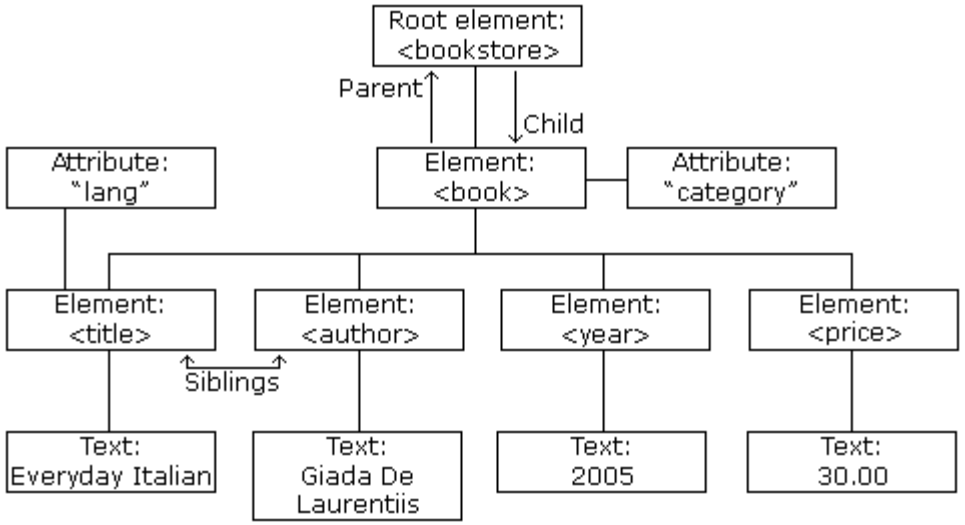
XML

- **Un documento XML** con sintassi corretta è chiamato "Well Formed". Per essere ben formato deve rispettare le seguenti regole:
  - Deve contenere **un unico elemento di massimo livello** (**root**) che contenga tutti gli altri. Fanno eccezione solo i commenti e le direttive di elaborazione;
  - Ogni elemento deve avere un **tag di chiusura**;
  - I tag di chiusura devono seguire l'ordine inverso ai rispettivi tag di apertura;
  - XML fa distinzione tra maiuscole e minuscole;

XML

Direttiva di elaborazione

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```



XML: Document Type Definition

Un documento XML, oltre a essere definito **Well Formed**, può anche essere definito **Valid** se convalidato rispetto a un **DTD (Document Type Definition)** o ad un **XML Schema**.

Un DTD definisce la struttura, gli elementi legali e gli attributi di un documento XML.

```
<!DOCTYPE note
[
    <!ELEMENT note (date,hour,to,from,body)>
    <!ELEMENT date (#PCDATA)>
    <!ELEMENT hour (#PCDATA)>
    <!ELEMENT to (#PCDATA)>
    <!ELEMENT from (#PCDATA)>
    <!ELEMENT body (#PCDATA)>
]>
```

```
<note>
  <date> 15-11-2023 </date>
  <hour>09:00</hour>
  <to> Studends </to>
  <from>IBM</from>
  <body>This is XML!</body>
</note>
```

XML: XML Schema

- **XML Schema** è un'alternativa basata su XML a DTD.

```
<xs:element name="note">
<xs:complexType>
  <xs:sequence>
    <xs:element name="date" type="xs:string"/>
    <xs:element name="hour" type="xs:string"/>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="body" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

```
<note>
  <date> 15-11-2023 </date>
  <hour>09:00</hour>
  <to> Studends </to>
  <from>IBM</from>
  <body>This is XML!</body>
</note>
```

XML: Differenze tra DTD e XMLSchema

DTD	XML Schema
Hanno poco in comune con la sintassi XML	Sono essi stessi degli XML
Non possono essere utilizzati gli stessi strumenti usati per lavorare con gli XML	Possono essere utilizzati gli stessi strumenti usati per lavorare con gli XML
Non consentono la definizione di tipi di dati. Pertanto non abbiamo controllo sul contenuto degli elementi di un XML	Consentono la definizione di tipi di dati. Abbiamo controllo sul contenuto degli elementi e sul valore degli attribute di un XML
Un XML dipende da un solo DTD	In un XML è previsto che tag diversi siano definiti in XML Schema diversi
Minore complessità	Maggiore complessità

Json: serializzazione e deserializzazione

- JSON sta per **JavaScript Object Notation**
- JSON è un formato di testo per **l'archiviazione** e **il trasporto di dati**;
- Il formato JSON è sintatticamente simile al codice per la creazione di oggetti JavaScript.
- JavaScript ha due funzioni incorporate per elaborare le stringhe:
  - **JSON.parse():** converte le stringhe JSON in oggetti JavaScript (**deserializzazione**);
  - **JSON.stringify():** converte un oggetto JavaScript in una stringa JSON (**serializzazione**);
- Anche in Java possiamo serializzare e **deserializzare** gli **oggetti**;
  - Utilizziamo specifiche **librarie** (es. Jackson, Gson);

Json: tipo di dati

- I dati sono in coppie nome/valore
- I dati sono separati da virgole
- Gli oggetti sono racchiusi tra parentesi graffe
- Gli array sono racchiusi tra parentesi quadre

```
{
  "name" : "Pippo",
  "age" : 100,
  "cartoon" : true,
  "friends" : [ "Pluto", "Paperino" ]
}
```

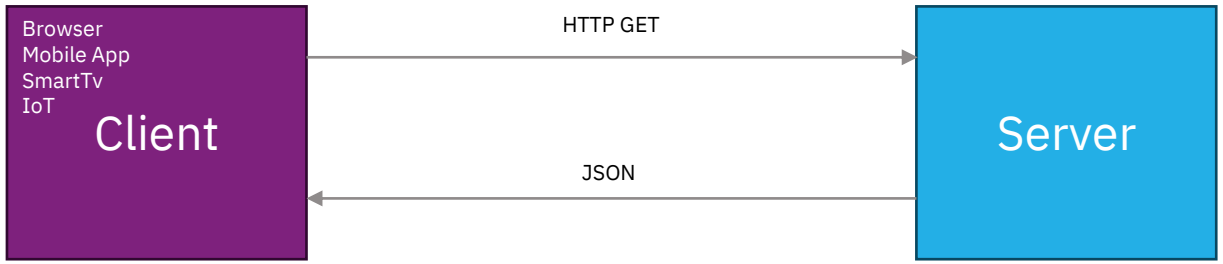
In **JSON** , i **valori** devono essere uno dei seguenti tipi di dati:

- stringa;
- numero;
- oggetto;
- array;
- booleano;
- null.

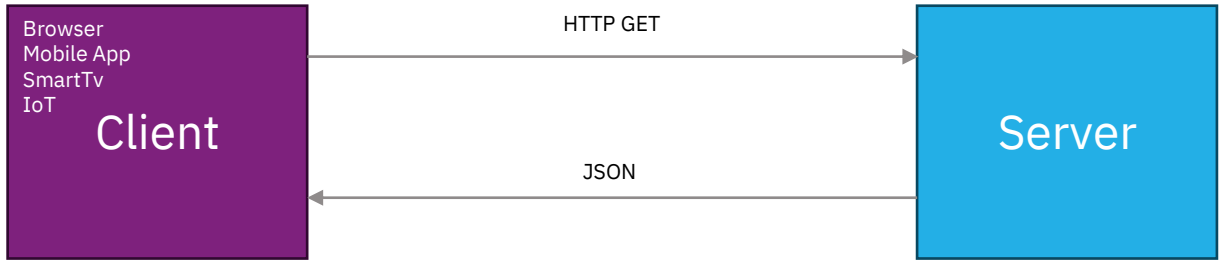
- **Undefined** **non** è un valore ammesso

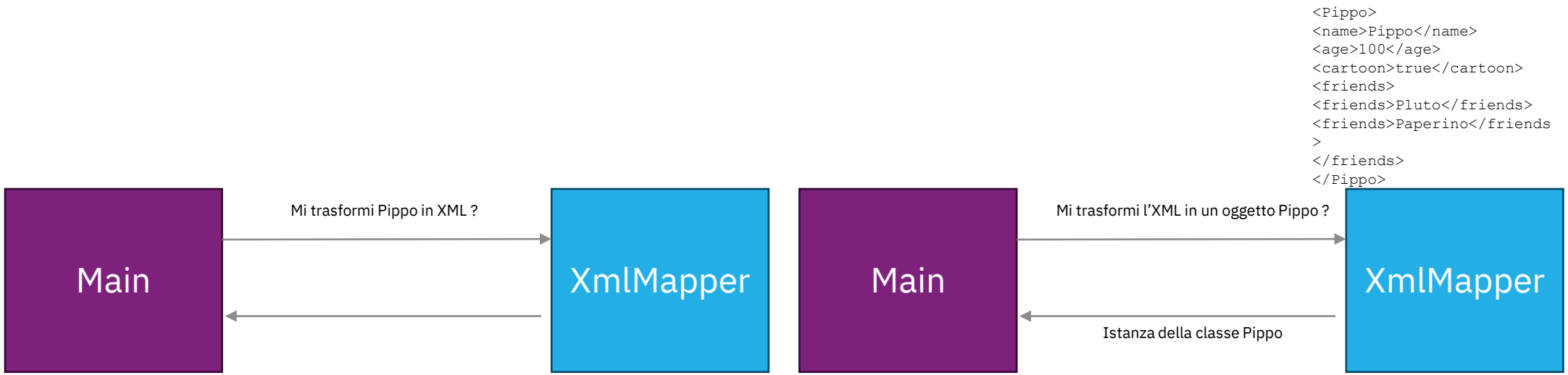
JSON: utilizzo

- Consento la conversione di **oggetti** Java in formato **JSON**;
- Utilizzati per la **comunicazione** tra servizi;
- Lo stile architetturale o anche definito paradigma **REST** si basa sullo scambio d'informazioni in formato JSON;
  - REST è l'acronimo di *REpresentation State Transfer* e si basa su protocollo HTTP;
  - Le famose **API REST** seguono questo paradigma;
- Medoto standard per la comunicazione client-server delle web-application moderne;



JSON: utilizzo





Differenze tra XML e JSON

XML	JSON
I dati XML sono senza tipo (dovrebbero essere tutte stringhe)	L’oggetto JSON ha un tipo (stringa, numero, booleano)
Supporta varie codifiche	Supporta solo la codifica UTF-8
Supporta i commenti	NON supporta i commenti
I dati XML devono essere analizzati	I dati sono facilmente accessibili come oggetti JSON
L’analisi XML cross-browser può essere complessa	È supportato dalla maggior parte dei browser
È più sicuro	È meno sicuro

Esercizio Java: Food Delivery

Creare un nuovo progetto **FoodDelivery** in quale consente:

- la creazione di un ordine;
- la cancellazione di un ordine;

Le **entità** che dovranno essere modellate sono le seguenti:

- **Ordine**: entità che rappresenta cosa il cliente vuole ordinare;
- **Restaurant**: che riceve gli ordini tramite un metodo **receiveOrder** dai clienti e li passa alla cucina per la prepartazione;
- **Kytchen**: che riceve gli orgini dal ristorante tramite un metodo **receiveOrder** e li **evade**;

Di seguito gli attributi che l'oggetto **Order** dovrà avere:

```
private int id;
private String orderElement;
private boolean readyToGo;
```

Quando un utente crea un nuovo ordine, specifica cosa vuole ordinare tramite l'attributo **orderElement**, l'ordine viene ricevuto dal ristorante che lo marca come non evaso tramite l'attributo **readyToGo** e lo smista alla cucina.
La cucina prende in carico l'ordine e lo marca come pronto impostando l'attributo **readyToGo**.

Testa la logica del programma con una classe **Test.java** che contiene un solo metodo main.