ITIS-LS "Francesco Giordani" Caserta

prof. Ennio Ranucci

a.s. 2019-2020

Semplici file XML JSON

Esercitazioni XML JSON

Introduzione

XML (eXtensible Markup Language) è un linguaggio che si basa sull'utilizzo di tag (es. <tag></tag>) simili a quelli utilizzati in HTML. L'idea di fondo è stata quella di creare un sistema con il quale fosse possibile trasportare dei dati indipendentemente dal programma utilizzato. XML utilizza infatti i tag per identificare il contenuto ed il tipo di dati da trasportare piuttosto che occuparsi di una loro forma di visualizzazione. La flessibilità di XML sta nel fatto di non avere dei tag predefiniti come ad esempio nel linguaggio HTML, ma possono essere creati in base alle proprie esigenze. In poche parole XML non fa nulla (da un punto di vista di esecuzione), ma dà la possibilità ai programmi di interpretarne in maniera corretta il suo contenuto.

Perché utilizzare

XML Nello sviluppo di pagine web possiamo sfruttare XML per dividere i dati effettivi dal linguaggio HTML. In questa maniera possiamo utilizzare HTML per concentrarci esclusivamente sul dove e come vengono visualizzati i dati. Con poche righe di codice JavaScript possiamo poi importare i dati necessari all'interno dei blocchi presenti nel codice HTML. XML non si limita solamente all'utilizzo nell'ambito del web, ma può essere utilizzato da programmi applicativi come strumento per esportare od importare dati tra versioni diverse dello stesso programma o addirittura da programmi completamente diversi. L'esportazione e l'importazione dei dati da un programma ad un altro può rivelarsi un lavoro molto dispendioso da un punto di vista di tempo. Le problematiche dipendono spesso e volentieri dai vari formati utilizzati per la memorizzazione dei dati. XML risolve questo problema utilizzando un sistema indipendente da una qualsiasi forma di hardware o software.

XML è stato progettato per trattare la descrizione delle informazioni, ossia concentra l'attenzione sul significato dei dati.
XML non è orientato ad uno specifico tipo di elaborazione.
XML è stato creato esclusivamente come uno strumento per strutturare, memorizzare e scambiare informazioni. Ad esempio, in questo documento XML che descrive un libro:

```
<?xml version="1.0"?>

<Libro>

   <Titolo>Cent'anni di solitudine</Titolo>

   <Autore>Gabriel Garcia Marquez</Autore>

   <Editore>Mondadori</Editore>

</Libro>
```

non è contenuta alcuna istruzione finalizzata all'elaborazione dei dati; si tratta di pura informazione confezionata nelle marcature XML. Occorre utilizzare un'applicazione specifica per inviare, ricevere o visualizzare il documento XML. Le marcature, evidenziando la semantica delle informazioni contenute in un documento, rendono semplice la progettazione e la realizzazione di applicazioni in grado di elaborare i dati.
XML non è stato progettato per sostituire HTML. I linguaggi XML e HTML sono stati progettati per svolgere compiti diversi:
XML è stato pensato per descrivere i dati ed è incentrato sul loro valore semantico;
XML riguarda la descrizione delle informazioni;
HTML (acronimo di Hyper Text Markup Language che significa linguaggio di marcatura per ipertesti) è stato pensato per visualizzare i dati ed è incentrato sull'aspetto con il quale il dato è presentato;
HTML tratta la visualizzazione delle informazioni.
Con XML è possibile condividere e scambiare dati in modo indipendente dall'hardware e dal software. Poiché i dati XML sono memorizzati in formato testuale piatto, ossia senza caratteri speciali aggiuntivi, XML fornisce un modo di condividere e scambiare i dati indipendente dal sistema operativo (Mac, Windows, UNIX) e dal software.

- *Un documento XML ècostituito da tre parti, che possono essere contenute in tre file diversi:*
  - *una Document Type Definition (DTD): èl'insieme delle regole che definisce la struttura dei dati, ovvero per la definizione degli elementi, degli attributi e delle loro correlazioni (la DTD definisce un vero linguaggio con un proprio vocabolario e regole sintattiche). Le DTD sono contenute alternativamente nel documento XML vero e proprio oppure in un file di testo con estensione dtd;*
  - *un foglio di stile: specifica le regole con cui un documento deve essere visualizzato da un applicativo (browser, word processor…);*
  - *il documento XML vero e proprio: i dati, organizzati secondo la struttura definita dalla DTD. I veri e propri documenti XML sono contenuti in file di solo testo con estensione xml.*
- *Come nel caso dell'HTML i dati possono alternativamente essere scritti in file oppure essere generati dinamicamente da applicativi quali i sistemi per la gestione di banche dati*

Un elemento è definito utilizzando la sintassi:
<!ELEMENT nome_elemento regola >
nome_elemento identifica un nuovo tag ed è una sequenza di caratteri che inizia con una lettera o un underscore e può poi contenere lettere, cifre, trattini, punti. Nel caso in cui si usino i namespace è anche possibile utilizzare ":"
Nel caso più semplice la regola può valere alternativamente: ANY oppure #PCDATA
ANY è una parola chiave che indica che fra <nome_elemento> e </nome_elemento> può essere contenuta qualsiasi cosa, ovvero possono essere contenuti sia dati (tipicamente del testo, "PCDATA"sta per parsed character data) sia altri elementi.
Se anziché ANY si utilizza #PCDATA non potranno essere contenuti altri elementi.

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es0*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Google Crome Versione 79.0.3945.79*
Obiettivo didattico:
Esempio di un documento dati XML
Obiettivo del programma:
Realizzare la struttura gerarchica seguente: lista comprende uno o più dati di tipo sito che, a loro volta, comprendono uno o più dati semplici nome, URL, descrizione e categoria.

Es0.DDT

```
<!DOCTYPE lista[
<!ELEMENT lista (sito+)>
 <!ELEMENT sito (nome, URL, descrizione, categoria)>
 <!ELEMENT nome (#PCDATA)>
 <!ELEMENT URL (#PCDATA)>
 <!ELEMENT descrizione (#PCDATA)>
 <!ELEMENT categoria (#PCDATA)>
<!ATTLIST sito IDSito ID #REQUIRED> ]>
```

Questo codice non contiene alcun dato, ma determina solamente la struttura dei dati che vengono rappresentati.
La clausola !DOCTYPE definisce l'oggetto lista, formato da vari elementi indicati con !ELEMENT. Il primo elemento lista è la radice della struttura che contiene elementi di tipo sito. Nella parentesi tonda, dopo il nome sito, è stato scritto il segno + per indicare che l'elemento lista può contenere uno o più elementi sito. L'elemento sito è formato da quattro dati indicati tra parentesi tonde.
Il tipo #PCDATA (Parsed Character Data) indica un dato di tipo testo (cioè formato da caratteri) che

può essere analizzato da un parser. Il parser, in generale, è un elaboratore di testi che possiedono una struttura sintattica predefinita. I parser XML elaborano un file XML per estrarne i dati passandoli a un browser Web oppure a un'applicazione. I moderni browser, come Internet Explorer, Firefox o Chrome, sono dotati di un parser che consente di visualizzare un file XML. La clausola !ATTLIST definisce un attributo per l'elemento sito: l'attributo è IDSito, di tipo ID, cioé il suo valore deve essere univoco all'interno del documento ed è un attributo obbligatorio (#REQUIRED). I valori degli attributi di tipo ID non possono iniziare con un numero. La definizione DOCTYPE del tipo di documento può essere inserita direttamente nel file XML oppure può essere salvata in un file separato con l'estensione .dtd. In questo secondo caso, il codice che rappresenta la struttura del documento è il seguente (non si scrive la parte !DOCTYPE):

Es0.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE lista SYSTEM "es0.dtd">
 <lista>
<sito IDSito="s001">
 <nome>Google</nome>
 <URL>http://www.google.it</URL>
 <descrizione>Il piu' famoso motore di ricerca</descrizione>
 <categoria>motori di ricerca</categoria>
</sito> <sito IDSito="s002">
<nome>La Repubblica</nome>
<URL>http://www.repubblica.it</URL>
 <descrizione>Versione on-line del quotidiano</descrizione>
 <categoria>news</categoria>
 </sito>
 </lista>
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es1*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Google Crome Versione 79.0.3945.79*
Obiettivo didattico:
XML Javascript Ajax
Obiettivo del programma:
Stampare a video il contenuto del file xml con javascript-Ajax

File leggiXml-htm

```
<html>

<head>

</head>

<body>

<div>

<b><span id="cognome0"></span> <span id="nome0"></span></b><br />

<span id="telefono0"></span><br />

<b><span id="cognome1"></span> <span id="nome1"></span></b><br />

<span id="telefono1"></span><br />

</div>

<script type="text/javascript">

if(window.XMLHttpRequest) {

//crea un parser in IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}else{

//crea un parser in IE5, IE6

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

xmlhttp.open("GET","rubrica.xml",false);

xmlhttp.send("");

xmlDoc=xmlhttp.responseXML;

var i;

for (i = 0; i < xmlDoc.getElementsByTagName("cognome").length; i++) {

  document.getElementById("cognome"+i).innerHTML=
```

```
xmlDoc.getElementsByTagName("cognome")[i].childNodes[0].nodeValue;
document.getElementById("nome"+i).innerHTML=
xmlDoc.getElementsByTagName("nome")[i].childNodes[0].nodeValue;
document.getElementById("telefono"+i).innerHTML=
xmlDoc.getElementsByTagName("telefono")[i].childNodes[0].nodeValue;

}
</script>
</body>
</html>
```

File rubrica.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE rubrica SYSTEM "rubrica.dtd">

<rubrica>
<contatto lingua="it">
  <cognome>Rossi</cognome>
  <nome>Mario</nome>
  <telefono>123456</telefono>
</contatto>
<contattolingua="it">
  <cognome>Verdi</cognome>
  <nome>Antonella</nome>
  <telefono>223344</telefono>
</contatto>
</rubrica>
```

<!ELEMENT ...>
Questo blocco identifica il nome dell'elemento da descrivere e tra parentesi tonde ne definisce la struttura. La struttura può essere definita come #PCDATA (combinazione mista di elementi, oppure un valore di testo), un elenco di sotto elementi presenti all'interno di esso, oppure una combinazione dei due tipi.
La presenza di un +, *, oppure ?, dietro al nome nell'elenco dei sotto elementi definisce rispettivamente se l'elemento deve essere presente una o più volte, mai o più volte, oppure mai o una volta.
<!ATTLIST ...>
Questo blocco permette di identificare le proprietà di un attributo all'interno di un elemento. La forma che utilizziamo è la seguente
<!ATTLIST *elemento attributo tipo_attributo {"default"} {stato}*>
elemento Il nome dell'elemento per il quale si sta definendo l'attributo
attributo Il nome dell'attributo che si sta definendo
tipo_attributo Come tipo di attributo possiamo trovare CDATA (che definisce come valore un testo), ID (che identifica un valore univoco all'interno del documento XML), IDREF (l'id utilizzato fa riferimento all'id di un altro elemento), IDREFS (che identifica un elenco di id utilizzate), NMTOKEN (identifica il nome di un elemento valido in XML), NMTOKENS (elenco di nomi XML), ENTITY (identifica un'entità definita), ENTITIES (un elenco di entità), NOTATION (il nome di una notazione), xml: (un valore predefinito di XML), oppure una serie di possibili valori all'interno di parentesi tonde separate dal simbolo pipe (|), es (si | no).
default Il valore di default nel caso in cui non venisse definito.
stato I valori possibili possono essere #REQUIRED (l'attributo deve essere sempre presente all'interno di ogni elemento), #IMPLIED (l'attributo è opzionale), oppure #FIXED *value* (il valore è fisso e non può essere modificato).

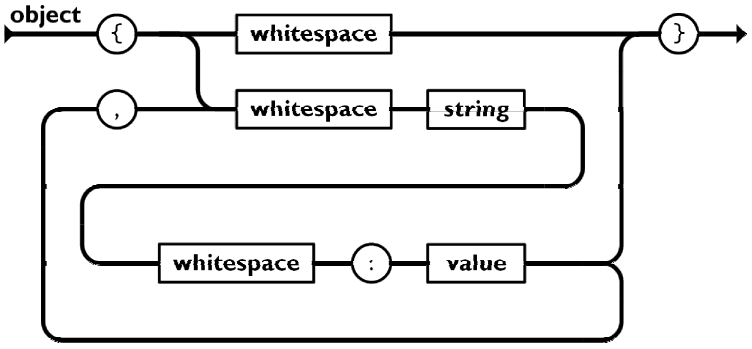
JSON

Json (Javascript Object notation), è un tipo di formato molto utilizzato per lo scambio dati in applicazioni client server come API o Mashup. È basato su JavaScript ma il suo sviluppo è specifico per lo scambio di dati ed è indipendente dallo sviluppo del linguaggio di scripting dal quale nasce e con il quale è perfettamente integrato e semplice da utilizzare.
Json è una valida alternativa al formato XML-XSLT e sempre più servizi di Web Services mettono a disposizione entrambe le possibilità di integrazione. Leggere ed interpretare uno stream in Json è semplice in tutti i linguaggi, non solo in JavaScript, con cui è completamente integrato ma anche con PHP, Java ed altri linguaggi server-side, per mostrare i dati da fonti esterne ed impaginarli secondo le proprie soluzioni personalizzate.

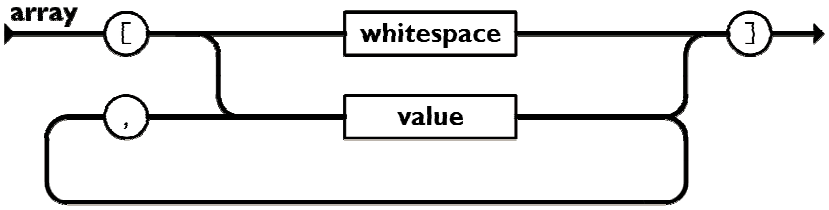
JSON è un formato di testo completamente indipendente dal linguaggio di programmazione, ma utilizza convenzioni conosciute dai programmatori di linguaggi della famiglia del C, come C, C++, C#, Java, JavaScript, Perl, Python, e molti altri. Questa caratteristica fa di JSON un linguaggio ideale per lo scambio di dati.
JSON è basato su due strutture:
Un insieme di coppie nome/valore. In diversi linguaggi, questo è realizzato come un oggetto, un record, uno struct, un dizionario, una tabella hash, un elenco di chiavi o un array associativo.Un elenco ordinato di valori. Nella maggior parte dei linguaggi questo si realizza con un array, un vettore, un elenco o una sequenza.Queste sono strutture di dati universali. Virtualmente tutti i linguaggi di programmazione moderni li supportano in entrambe le forme. E' sensato che un formato di dati che è interscambiabile con linguaggi di programmazione debba essere basato su queste strutture.

In JSON, assumono queste forme:

Un oggetto è una serie non ordinata di nomi/valori. Un oggetto inizia con {parentesi graffa sinistra e finisce con }parentesi graffa destra. Ogni nome è seguito da :due punti e la coppia di nome/valore sono separata da ,virgola.

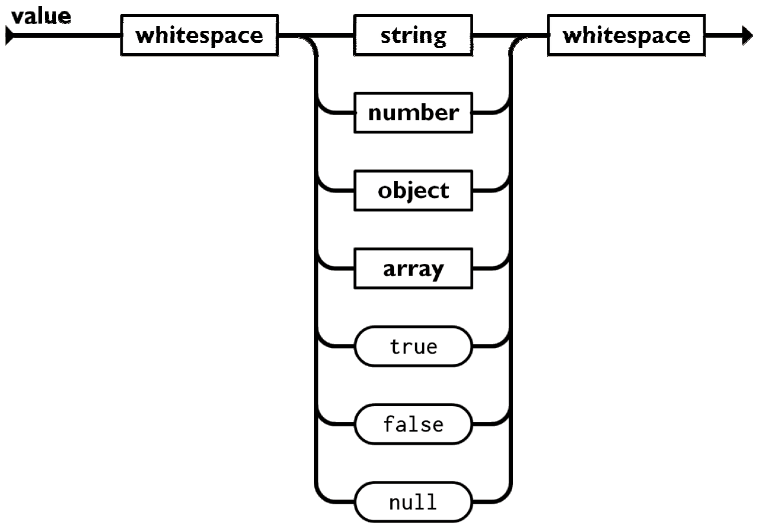


Un array è una raccolta ordinata di valori. Un array comincia con [parentesi quadra sinistra e finisce con]parentesi quadra destra. I valori sono separati da ,virgola.

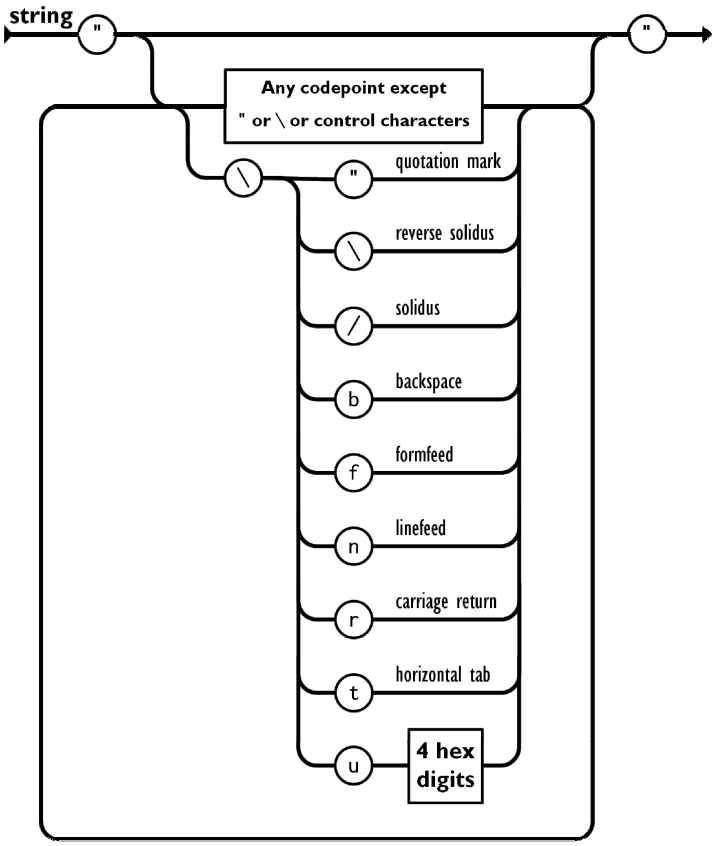


Un valore può essere una stringa tra virgolette, o un numero, o vero true o falso false o nullo null, o un oggetto o un array. Queste strutture possono essere annidate.

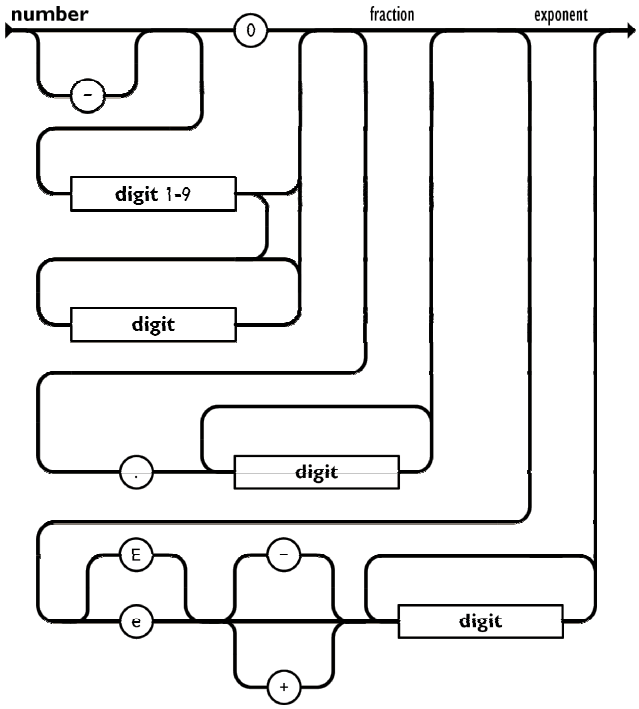
Una stringa è una raccolta di zero o più caratteri Unicode, tra virgolette; per le sequenze di escape



utilizza la barra rovesciata. Un singolo carattere è rappresentato come una stringa di caratteri di lunghezza uno. Una stringa è molto simile ad una stringa C o Java.



Un numero è molto simile ad un numero C o Java, a parte il fatto che i formati ottali e esadecimali non sono utilizzati.



ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es2*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Google Crome Versione 79.0.3945.79*
Obiettivo didattico:
JSON Javascript Ajax
Obiettivo del programma:
Inviare unstringa JSON ad un file .php

```
<!DOCTYPE html>
<html>
<body>
<h2>Convert a JavaScript object into a JSON string, and send it to the server.</h2>
<script>
var myObj = { name: "John", age: 31, city: "New York" };
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
</script>
</body>
</html>
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es3*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Google Crome Versione 79.0.3945.79*
Obiettivo didattico:
JSON Javascript Ajax
Obiettivo del programma:
Convertire una stringa JSON in un oggetto Javascript

```
<!DOCTYPE html>
<html>
<body>
<h2>Convert a string written in JSON format, into a JavaScript object.</h2>

<p id="demo"></p>

<script>

var myJSON = '{"name":"John", "age":31, "city":"New York"}';

var myObj = JSON.parse(myJSON);

document.getElementById("demo").innerHTML = myObj.name;

</script>
</body>
</html>
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es4*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Google Crome Versione 79.0.3945.79*
Obiettivo didattico:
JSON Javascript Ajax
Obiettivo del programma:
Scrivere e leggere una stringa JSON

```
<!DOCTYPE html>

<html>

<body>


<h2>Store and retrieve data from local storage.</h2>


<p id="demo"></p>


<script>

var myObj, myJSON, text, obj;


// Storing data:

myObj = { name: "John", age: 31, city: "New York" };

myJSON = JSON.stringify(myObj);

localStorage.setItem("testJSON", myJSON);


// Retrieving data:

text = localStorage.getItem("testJSON");

obj = JSON.parse(text);

document.getElementById("demo").innerHTML = obj.name;

</script>


</body>

</html>
```