Università Degli Studi di Ferrara

Corso di Laurea in Informatica - A.A. 2023 - 2024

Tecnologie Web

Lez. 10 - XML

Tecnologie Web - Docente: Giacomo Piva - giacomo.piva@unife.it

Cap. 4

Tecnologie Client Side

In questa lezione…

- XML
- AJAX
- JSON

Il linguaggio XML

E**X**tensible **M**arkup **L**anguage

Derivato da Standard Generalized Markup Language (SGML)

Progettato nel 1998 per la pubblicazione di dati in formato elettronico su larga scala

E’ un W3C Standard

XML vs HTML

Nascono e si sviluppano con due scopi ben distinti:

HTML: Orientato alla visualizzazione e alla formattazione di documenti (come appaiono i dati)

XML: Orientato al trasporto dei dati, descrive la struttura del documento ed è utilizzato per scambiare i dati (cosa sono i dati)

Il metalinguaggio XML

Alcuni cenni sulla sintassi del linguaggio XML

- La prima linea deve contenere la versione di XML
- XML è case sensitive
- Tutti i tag vanno sempre chiusi
- I tag devono essere annidati in maniera corretta
- I tag devono essere in lowercase
- Deve sempre esistere un solo nodo radice
- Gli attributi vanno sempre specificati fra apici (singoli o doppi)

Un Esempio

Supponiamo di sviluppare un semplicissimo gestore di prodotti e che il seguente sia la porzione di documento XML che descrive un prodotto:

```
<?xml version="1.0" ?>
<product>
    <name>Nike Air</name>
   <image>https://via.placeholder.com/800x600.jpg</image>
    <price>60.00</price>
</product>
```

La sua comprensione è banale, XML è studiato per essere autoesplicativo

Il metalinguaggio XML

XML è un meta-linguaggio di markup

È bene precisare che non esistono dei tag XML predefiniti, i tag vengono definiti dall'applicazione (cioè dalle nostre necessità) e la formalizzazione della sintassi viene definita attraverso due entità:

- DTD (Document Type Definition)
- XSD (XML Schema Definition)

XML Schema (XSD)

I tag vengono definiti in base all'applicazione, in pratica non ci sono tag predefiniti come nell'HTML.

La funzione di definire formalmente gli elementi e la struttura di un documento XML è delegata agli XML Schema (XSD).

All'interno di XML Schema vengono definiti:

- Gli elementi e gli attributi che appaiono nel documento XML;
- I figli degli elementi (sia il loro numero che il loro ordine);
- Quando un elemento può essere vuoto e quando deve contenere testo;
- Il tipo degli elementi e degli attributi;
- Il valore di default degli attributi di un elemento.

Tipi di dato XSD

I tipi di dato predefiniti sono 6:

- String
- Decimal
- Integer
- Boolean
- Date
- Time

Ma è possibile definire dei tipi di dato complessi (complex type) per creare nuovi tipi e riutilizzarli all’interno di altri file XSD

Correttezza dell’XML

E’ necessario fare un'importante distinzione quando ci si riferisce alla “correttezza” di documento XML

Un documento XML è:

- Ben formattato: se rispetta le regole sintattiche di base
- Valido: Se rispetta le regole descritte all’interno dello schema XSD di riferimento

La validazione avviene attraverso appositi tool (es. xmllint) o attraverso librerie (es. libXML2)

Esempio

Riguardando l’esempio di prima …

```
<?xml version=“1.0” ?>
<product>
    <name>Nike Air</name>
    <image>http://placehold.it/800x600.jpg</image>
    <price>60.00</price>
</product>
```

Esempio

Con le poche informazioni in nostro possesso fino a ora, possiamo già facilmente intuire lo schema XSD che lo definisce

```
<?xml version=“1.0” ?>
<schema>
    <element name="product">
        <complexType>
            <sequence>
                <element name="name" type=”string"/>
                <element name="image" type=“string"/>
                <element name="price" type=“decimal"/>
            </sequence>
        </complexType>
    </element>
</schema>
```

XSD - XML Schema Definition

Il linguaggio per definire i file XSD è molto articolato, noi ci limiteremo a capire il funzionamento della validazione di un file XML.

Per approfondire il linguaggio potete fare riferimento al sito: https://www.w3schools.com/xml/schema_intro.asp

Esempio XML

Validazione XML

Namespace

Un Namespace è un’insieme di nomi di entità il cui scopo è quello di evitare ambiguità nel caso facendo riferimento a un nome sia possibile riferirsi a più entità.

Pensiamo al caso in cui il software che stiamo realizzando faccia uso di 2 librerie: una per la gestione di grafici e una per gestire le posizioni geografiche. È molto probabile che entrambe queste librerie implementino una classe chiamata `“Point”`.

Namespace negli XSD

Non essendoci tag predefiniti ed essendo possibile riutilizzare gli schemi XSD diventa necessario dover far fronte a delle possibili ambiguità, ovvero distinguere fra più tag che potrebbero avere lo stesso nome (es. `“table”`).

I Namespace risolvono gli eventuali conflitti rendendo non ambiguo ogni elemento (o attributo) facendolo precedere da un prefisso, come ad esempio `<xs:table>`.

L’utilizzo dei Namespace è un W3C Recommendation

Namespace negli XSD

I Namespace si introducono in un documento XSD attraverso l'elemento `schema` e l'attributo `xmlns` il cui valore è un URL che ha valore solamente informativo.

```
<schema xmlns:xs="http://www.w3.org/2001/XMLSchema“>
```

Per rendere non ambiguo ogni elemento è necessario quindi farlo precedere dal prefisso del Namespace di riferimento ad esempio:

```
<xs:element name=“fname” type=“string”>
```

Esempio

Il nostro esempio senza ambiguità risulterà essere:

```
<?xml version=“1.0” ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="product">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="name" type=”xs:string"/>
                <xs:element name="image" type=“xs:string"/>
                <xs:element name="price" type=“xs:decimal"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Esempio Namespace

Validazione XML Namespace

AJAX

Asynchronous JavaScript and XML

Un insieme di tecnologie utilizzate per lo scambio di dati tra client (browser) e il server.

- Oggetto XMLHttpRequest per scambiare I dati;
- JavaScript per mostrare i dati;
- XML (ma non solo) formattato usato per trasportare i dati.

Non è un W3C Recommendation

AJAX

Documento HTML

XMLHttpRequest

XML

XMLHttpRequest

È appunto un oggetto, in quanto ha metodi e proprietà, ma la sua istanziazione …

CAMBIA A SECONDA DEL BROWSER

XMLHttpRequest

Proprietà:

- **`readyState`**: contiene lo stato dell'oggetto. Può assumere 5 valori:

  0 = Uninitialized: l'oggetto esiste ma non è stato inizializzato;
  1 = Open: l'oggetto è aperto;
  2 = Sent: la richiesta è stata inviata;
  3 = Receiving: i dati stanno arrivando a destinazione;
  4 = Loaded: operazione completata.

- **`responseText`**: contiene il risultato di una richiesta in formato testuale

- **`responseXML`**: Contiene il risultato di una richiesta in formato XML

- **`status`**: Contiene lo stato di una richiesta (200, 404, 503 …)

XMLHttpRequest

Metodi:

- **`abort()`**: termina preventivamente la richiesta HTTP.
- **`getResponseHeaders()`**: restituisce le intestazioni della risposta.
- **`open()`**: apre la richiesta.
- **`send()`**: invia la richiesta.
- **`setRequestHeader()`**: imposta le intestazioni della richiesta.

XMLHttpRequest

```
// Istanzio un oggetto XMLHttpRequest
// Andrebbero gestiti tutti i browser
var myRequest = new XMLHttpRequest();

// Definisco cosa fare quando lo stato cambia
// Gli passo una funzione che poi gestirò ...
myRequest.onreadystatechange = StateChanged();

// Preparo l'URL con le variabili
var url = "myscript.php?nome=Giovanni";

// Setto l'header della richiesta
myRequest.open("GET", url, true);

// Invio la richiesta includendo il body
// in questo caso nessun body
myRequest.send(null);
```

XMLHttpRequest

```
function StateChanged(request) {
    // Quando lo stato è "complete"
    if (request.readyState == 4) {
        // Controllo che la risposta sia OK
        if (request.status == 200) {
            // Visualizzo I risultati nel documento
            document.write(request.responseXML);
        } else {
            // Se la risposta non è OK visualizzo un alert
            alert ('Errore: '+ request.status);
        }
    }
}
```

AJAX con jQuery

Per fortuna esistono le librerie JavaScript che astraendo il linguaggio ci facilitano la gestione del linguaggio nei vari browser.

In particolare jQuery ha un set di funzioni per poter usare AJAX con facilità.

Vedi Lezione 9 - JavaScript, slide nr. 31

AJAX con jQuery

Il metodo per una chiamata AJAX più flessibile è il metodo `ajax()` che consente di specificare moltissimi parametri compreso funzioni da eseguire prima di effettuare la chiamata `beforeSend()`

Noi vedremo solo le configurazioni essenziali:

- `url` `// URL da chiamare`
- `metodo` `// metodo http GET, POST …`
- `dataType` `// tipo di dato di ritorno`
- `data` `// opzionale`

AJAX con jQuery

```
$.ajax({
      url: “myscript.php”,
      type: "GET",
      dataType: ”xml”,
      data: “id=11”,
      success: function(data) {
            // Il mio codice success
      },
      error: function(response, status) {
            // Il mio codice error
      }
});
```

Esempio 1

AJAX con jQuery

AJAX con jQuery

Una volta ottenuta la risposta in formato XML, bisogna effettuarne il parsing per poter estrarre i dati.

jQuery è di natura in grado di eseguire il parsing di un XML e di estrarne i contenuti. Il parsing del documento XML è comunque un passaggio di cui si preferirebbe fare a meno.

Esempio 2

XML Parsing con jQuery

XML e il Web

Nonostante XML e il Web siano fortemente legati e alcune tecnologie sembrino indissolubilmente legate a esso, ci sono altre tecnologie che sono diventate preferibili a XML nell'utilizzo sul Web.

XML ha "perso" il confronto con altre tecnologie che hanno lo stesso scopo (cioè trasportare i dati) che lo hanno superato in quanto più flessibili e meno verbose e spesso più "adatte".

JSON

Java**S**cript **O**bject **N**otation

JSON è una sintassi usata soprattutto sul Web per immagazzinare e trasportare dati.

È preferita ad XML soprattutto per 3 motivi:

- Più semplice: è indiscutibilmente più semplice da comprendere
- Più flessibile: non richiede formalismi (es. XSD)
- Meno verboso: vediamo un esempio

JSON vs XML

```
<?xml version="1.0" ?>
<product>
    <name>Nike Air</name>
    <image>http:…</image>
    <price>60.00</price>
</product>
```

```
data: {
    "product": {
        "name": "Nike Air",
        "image": "http://…",
        "price": 60.00
    }
}
```

	Dimensione	Costo
Informazione	20 caratteri	
XML	102 caratteri	82 caratteri (82 Byte)
JSON	68 caratteri	48 caratteri (48 Byte)

Sintassi JSON

La sintassi JSON è costituita da coppie

`“nome":”valore”`

I valori possono essere:

- Stringhe
- Numbers
- Array
- Set (insiemi, gruppi, ecc…)

Sintassi JSON

Stringhe: `nome:”stringa”`

Interi: `“nome”:22`

Floats: `“nome":22.4`

Array: `“nome”: [11, 12, 13, …]`

Set: `“nome": { “nome”:”stringa”, “nome2”:22, … }`

Sintassi JSON

È possibile strutturare l'informazione come un oggetto JSON:

```
"orologio": {
    "marca": "Nautica",
    "modello": "NA346-T"
    "image": "http://…",
    "price": 60.00,
    "colours": ["bianco", "blu"]
}
```

Come sempre, potete approfondire sul sito W3School all'indirizzo: https://www.w3schools.com/js/js_json_syntax.asp

JSON in PHP

È possibile convertire un array associativo in un JSON e viceversa con le funzioni

```
json_encode();
json_decode();
```

La prima funzione restituisce come valore di ritorno l'array associativo passato come parametro serializzato in una stringa JSON

La seconda esegue l'operazione inversa

JSON in PHP

```
$json = json_encode($array);
```

```
$array = json_decode($string);
```

JSON in JavaScript

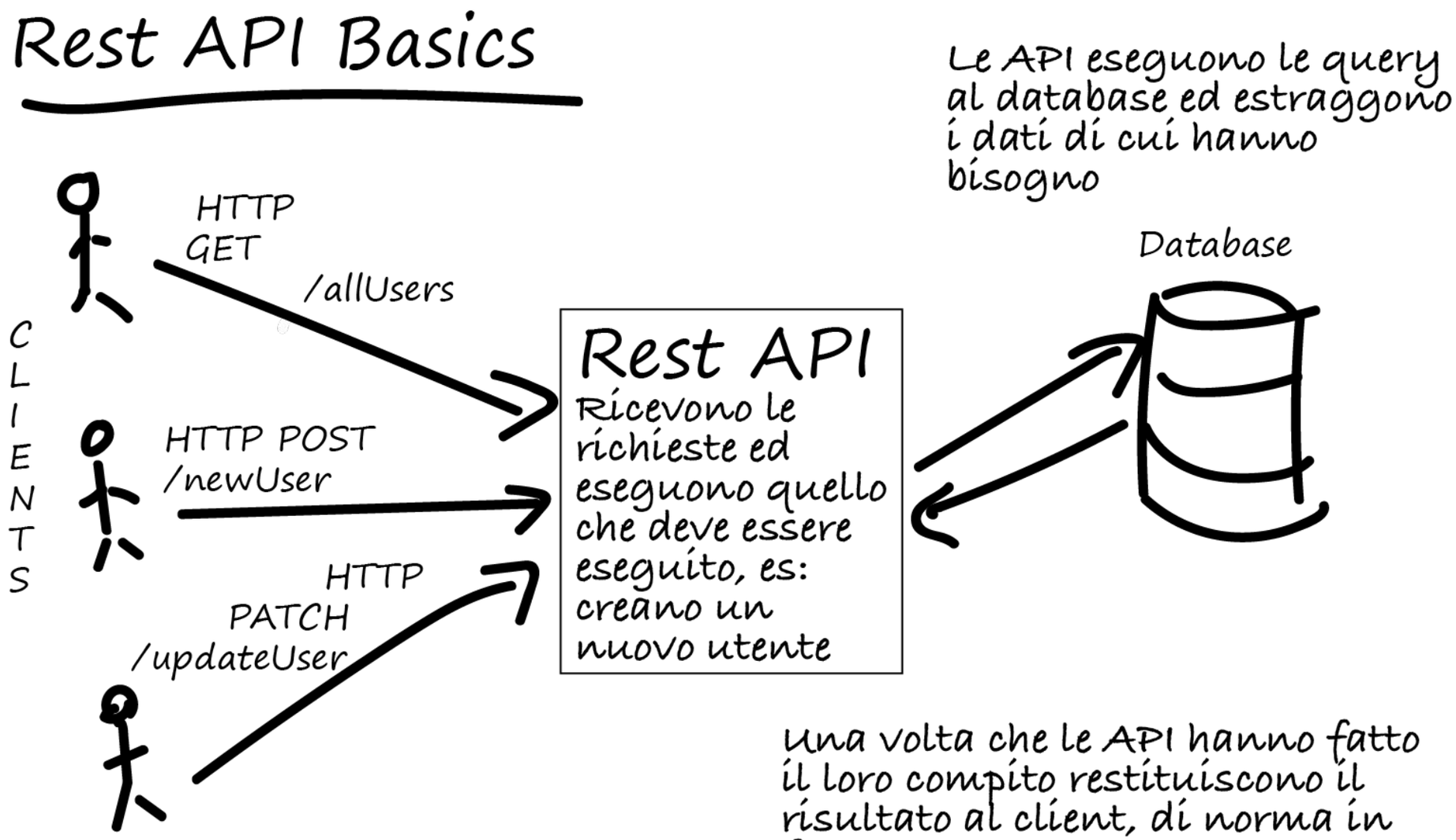
Una volta ottenuti i dati in formato JSON all'interno del nostro codice possiamo usarli con un oggetto JavaScript o come un array di oggetti

Vedi Lezione 9 - JavaScript, slide nr. 21

Esempio 3

JSON encode/decode

API REST


Rest API Basics
HTTP
GET
/allUsers
CLIENTS
HTTP POST
/newUser
HTTP
PATCH
/updateUser
Rest API
Ricevono le richieste ed eseguono quello che deve essere eseguito, es: creano un nuovo utente
Database
I client inviano le richieste ed attendono le risposte
Le API eseguono le query al database ed estraggono i dati di cui hanno bisogno
Una volta che le API hanno fatto il loro compito restituiscono il risultato al client, di norma in formato XML o JSON

Esempio 4

REST e JSON

Domande?