6. XML E RAPPRESENTAZIONE DEL FORMATO DI RISPOSTA

6.1 INTRODUZIONE

Nel presente capito, che trae spunto dai contenuti presenti nel tutorial online del W3Schools [S13], è esposta una breve introduzione al linguaggio XML. Ciò viene fatto allo scopo di poter meglio comprendere le informazioni presenti nei record bibliografici, che vengono comunemente scambiati nel formato XML, così come definito nel protocollo OAI-PMH.
E' bene precisare che le considerazioni fatte sull'argomento sono relative all'utilizzo dello Schema XML per la validazione dei documenti, senza considerare il DTD, che ormai non è più largamente usato per le validazioni.
Alla fine del capitolo, è presentato il formato XML di risposta relativo al protocollo OAI-PMH [S14].

6.2 LINGUAGGIO ESTENSIBILE XML

XML è l'acronimo di eXtensible Markup Language, un linguaggio di contrassegno (markup) simile ad HTML, progettato per la descrizione dei dati.

XML ed HTML differiscono per i seguenti aspetti:

- I tag XML non sono stabiliti a priori, ma devono essere definiti dall'utente, a differenza di quelli HTML
- XML fu progettato per il trasporto dei dati
- HTML ha lo scopo principale di visualizzare le informazioni, mentre XML di descrivere le informazioni.

Esso può utilizzare per descrivere i dati i seguenti metodi:

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

Entrambi sono stati progettati per la validazione di documenti XML e per renderli autodescrittivi.
Attualmente, XML Schema è il metodo che si sta diffondendo maggiormente perché offre maggiori vantaggi rispetto al DTD, tra i quali:

- È estensibile per aggiunte future
- È più ricco e più utile del DTD
- È scritto anch'esso in XML
- Supporta vari tipi di dati
- Supporta il namespace (spazio dei nomi)

Per maggiori approfondimenti sullo schema XML vedi paragrafo 6.8.

Una delle caratteristiche più importanti di XML è l'estensibilità, cioè la possibilità di definire i propri tag. Mentre HTML, ad esempio, ha dei tag predefiniti: <b>, <table>, <h1>, ecc., XML permette all'autore di definire i tag e la struttura del documento, come nel seguente esempio:

```
<appunto>
  <da>Basilio</da>
  <a>Ercole</a>
  <oggetto>Conferma</oggetto>
  <body>Appuntamento confermato per le ore 16:00.</body>
</appunto>
```

Infatti i tag dell'esempio precedente, come <appunto>, <da>, <a>, non sono definiti in XML.

XML è ormai divenuto uno "strumento" importante per il Web e si può prevedere già da adesso come nel prossimo futuro esso sarà comunemente utilizzato per la manipolazione, l'immagazzinamento e la trasmissione dei dati.

6.3 USI DI XML

Differenti sono le applicazioni in cui XML viene comunemente impiegato, tra le quali le seguenti:

- Integrazione tra XML e HTML
- Scambio dei dati tra sistemi incompatibili
- Condivisione dei dati
- Immagazzinamento dei dati
- Accessibilità dei dati

Tali aspetti verranno discussi nei paragrafi seguenti.

6.3.1 Integrazione tra XML e HTML

Un documento HTML semplice può contenere al suo interno dei dati che vengono visualizzati in base ai tag utilizzati. Con XML i dati possono essere immagazzinati su file XML separati dal documento HTML. In tal modo qualunque modifica sui dati non comporta modifiche corrispondenti nel documento HTML e viceversa per quel che riguarda la variazione sulla formattazione. In tal caso, la sintassi da utilizzare è la seguente:

```
<xml id="appunto" src="appunto.xml">
```

dove **appunto.xml** è il file XML separato, contenente le informazioni.

E' anche possibile immagazzinare i dati dentro il file HTML stesso, come "isole di dati", fermo restando che modifiche all'interno delle isole non ne comportano nella parte relativa alla visualizzazione, e viceversa. In tal caso, la sintassi da utilizzare è la seguente:

```
<xml id="appunto">
<appunto>
  <da>Basilio</da>
  <a>Ercole</a>
  <oggetto>Conferma</oggetto>
  <body>Appuntamento confermato per le ore 16:00.</body>
</appunto>
</xml>
```

6.3.2 Scambio dei dati tra sistemi incompatibili

I sistemi di computer e i database contengono dati in formati incompatibili. Uno

dei problemi che interessa gli implementatori di tali sistemi è quello dello scambio dei dati attraverso Internet. La conversione di essi in XML può essere una soluzione, in quanto vengono creati dei dati che possono essere letti da molti tipi differenti di applicazioni.

6.3.3 Condivisione dei dati

I dati in XML vengono immagazzinati nel formato di testo semplice. Ciò costituisce un modo indipendente dal software e dall'harware per la condivisione e rende più facile creare dati che possono essere trattati da diverse applicazioni.

6.3.4 Immagazzinamento dei dati

Con XML è possibile anche immagazzinare i dati in file di testo semplice o in database. Infatti esistono delle applicazioni in grado di memorizzare e recuperare informazioni da un file o da un database, ed altre che possono essere usate per visualizzare le stesse informazioni.

6.3.5 Accessibilità dei dati

XML è indipendente da hardware, software ed applicazioni. In tal modo è possibile rendere disponibili i propri dati ad altre applicazioni client oltre che ai browser HTML.
I dati possono dunque essere utilizzati da tutti i tipi di "macchine lettrici", ovvero qualunque tipo di dispositivo in grado di presentare tali informazioni alle persone disabili (come ad esempio le persone affette da disturbi visivi).

6.4 SINTASSI XML

Si definisce documento XML ben formato qualunque documento che rispetta le regole sintattiche del linguaggio.

La sintassi di XML ha le seguenti regole:

- Tutti gli elementi XML devono avere un tag di chiusura
- I tag XML sono case sensitive
- Tutti gli elementi XML devono essere propriamente nidificati
- Tutti i documenti XML devono avere un elemento radice
- I valori degli attributi devono essere sempre racchiusi tra apici
- Gli spazi bianchi vengono mantenuti
- CR / LF è convertito in LF
- E' possibile inserire dei commenti

Si definisce documento XML valido, un documento XML ben formato, che è conforme alle regole di uno schema XML.

Per approfondimenti sullo *schema XML* vedere il paragrafo 6.8

La sintassi XML è semplice ed autodescrittiva, come si può vedere nel seguente esempio:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<appunto>
  <da>Basilio</da>
```

```
  <a>Ercole</a>
  <oggetto>Conferma</oggetto>
  <body>Appuntamento confermato per le ore 16:00.</body>
</appunto>
```

La prima linea dell'esempio rappresenta la dichiarazione XML, che indica la versione (1.0) e la codifica (ISO-8859-1) utilizzata nel documento.

La linea successiva individua l'elemento radice che specifica in un certo senso cosa il documento rappresenta (un appunto).

Le 4 linee successive indicano gli elementi figli dell'elemento radice: da, a, oggetto, body.

L'ultima linea definisce il tag di chiusura dell'elemento radice.

6.4.1 Tag di chiusura per gli elementi XML

Tutti gli elementi XML devono avere per ogni tag specificato, uno corrispettivo di chiusura.
Mentre in HTML ciò non è obbligatorio, come per esempio:

```
<br>Questa è una nuova riga
<p>Questo è un nuovo paragrafo
```

viceversa in XML è illegale omettere il tag di chiusura.
Gli unici tag che fanno eccezione a questa regola sono quelli di dichiarazione, in quanto essa non fa parte del documento XML stesso. Ad esempio:

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

6.4.2 Tag XML case sensitive

XML, contrariamente ad HTML, è case sensitive, dunque i tag devono rispettare tale convenzione, quindi due di essi, contenenti le stesse lettere, ma che differiscono per le maiuscole e minuscole vengono considerati differenti, come per esempio <Oggetto> ed <oggetto>.

Conseguentemente i tag di apertura e chiusura devono rispettare le stesse maiuscole e minuscole.

6.4.3 Annidamento degli elementi XML

A differenza di HTML, dove è consentito annidare gli elementi in maniera impropria come ad esempio:

```
<b><i>Questo testo è grassetto e corsivo</b></i>
```

in XML ciò non è consentito, in quanto tutti gli elementi devono essere annidati uno dentro l'altro. Quindi in XML si avrebbe invece:

```
<b><i>Questo testo è grassetto e corsivo</i></b>
```

6.4.4 Elemento radice

Ogni documento XML deve contenere obbligatoriamente un tag radice che deve

essere necessariamente chiuso alla fine. Tale tag conterrà gli elementi figli al suo interno i quali a loro volta potranno contenerne altri correttamente annidati.

6.4.5 Utilizzo degli apici

Gli elementi XML possono contenere coppie nome-valore all'interno dei tag che vengono detti *attributi*. I valori di tali attributi devono sempre essere racchiusi tra apici, come ad esempio:

```
<appuntamento data="27/05/04" ora="16:38">
```

Vedi paragrafo 6.6 riguardo gli attributi XML.

6.4.6 Spazi bianchi

Con XML, a differenza di quanto avviene con HTML, gli spazi bianchi all'interno di un documento non verranno troncati.
Difatti una linea del tipo:

```
<body>Appuntamento confermato per le ore              18:30</body>
```

sarà visualizzata in HTML nel seguente modo:

```
Appuntamento confermato per le ore 18:30
```

Mentre in XML manterrà la forma:

```
Appuntamento confermato per le ore              18:30
```

6.4.7 Rappresentazione del carattere di una nuova linea

Nelle applicazioni Windows, un new line viene normalmente rappresentato con la coppia di caratteri CR / LF (carriage return / line feed), viceversa in applicazioni Macintosh viene utilizzato solo il carattere CR, mentre in Unix si adopera solo LF.
Con XML un new line deve essere sempre specificato come LF.

6.4.8 Commenti in XML

Così come HTML, anche XML, permette di poter scrivere all'interno di un documento delle linee di commento. La sintassi è la stessa per entrambi e cioè consiste nel racchiudere il commento tra i seguenti delimitatori:
"<!--" e "-->".

6.5 ELEMENTI XML

Un elemento XML è tutto ciò che va da un tag di apertura al corrispettivo tag di chiusura compreso il contenuto all'interno.

Il contenuto di un elemento XML può essere di tre tipi:

- *Misto* – contiene sia testo che altri elementi:

```
<capitolo>Introduzione a XML
   <paragrafo>Cos'è HTML</paragrafo>
   <paragrafo>Cos'è XML</paragrafo>
```

```
</capitolo>
```

- *Semplice* – contiene solo testo:
  ```
  <title>My First XML</title>
  ```
- *Vuoto* – non contiene informazioni al suo interno:
  ```
  <prod id="33-657" media="paper"></prod>
  ```

Come l'esempio appena visto, un elemento può contenere degli attributi.

Gli elementi XML sono *estendibili*, cioè un documento XML preesistente può essere arricchito successivamente con l'aggiunta di nuovi elementi. In tal modo, un'applicazione che fa uso della prima versione di tale documento, sarà ancora in grado di trattare il nuovo documento, perché ignorerà i nuovi tag aggiunti.

Per l'assegnazione dei nomi agli elementi XML devono essere seguite le seguenti regole:

- I nomi possono contenere lettere, numeri e altri caratteri
- I nomi non devono iniziare con un numero o con un carattere di punteggiatura
- I nomi non devono iniziare con le lettere xml o con le varianti maiuscole e minuscole (XmL, xMl, …)
- I nomi non possono contenere spazi

E' buona norma comunque adottare anche i seguenti suggerimenti:

- Usare dei nomi descrittivi: non ci sono parole riservate
- E' possibile usare l'underscore per i nomi composti: <titolo_libro>, <titolo_capitolo>
- Evitare l'utilizzo dei simboli "-" e "." per i nomi composti: il primo è

ambiguo poiché alcuni software potrebbero interpretarlo come simbolo di sottrazione, mentre il secondo potrebbe essere interpretato come simbolo concatenatore di stringhe.

- Anche se non è presente alcuna limitazione riguardo la lunghezza dei nomi, bisogna evitare di usarne di troppo lunghi.
- Evitare di usare lettere accentate quali ad esempio è, à, ò perché alcuni software non li supportano.
- Il carattere ":" non dovrebbe essere usato nei nomi degli elementi perché il suo utilizzo è riservato ai namespace (vedi paragrafo 6.7)

6.6 ATTRIBUTI XML

Un attributo XML è un mezzo per fornire informazioni aggiuntive circa l'elemento in cui esso è specificato.

L'utilizzo degli attributi in XML è analogo a quello dell'HTML, in quanto consiste nell'uso di una coppia nome-valore accanto all'elemento che si vuole specificare, come nel seguente esempio:

```
<file tipo="gif">computer.gif</file>
```

Benchè l'attributo **tipo** in questione non aggiunga niente al dato contenuto nell'elemento **file**, esso contribuisce a fornire informazioni che ad esempio potrebbero essere necessarie ad un software che vuole manipolare l'elemento stesso.

I valori che vengono assegnati agli attributi devono essere sempre racchiusi tra

virgolette singole o doppie.
Non esistono regole di preferenza all'una o all'altra notazione, l'unica accortezza da seguire è che se il valore dell'attributo contiene al suo interno delle virgolette singole, bisognerà racchiuderlo tra virgolette doppie e viceversa, se contiene virgolette doppie.

```
<giocatori nome="Roberto 'Codino' Baggio">
<giocatori nome='Roberto "Codino" Baggio'>
```

A differenza dell'HTML, l'uso massiccio di attributi in XML non è raccomandabile, mentre è da preferirsi l'utilizzo di elementi per specificare i dati nel documento.
I motivi per cui bisognerebbe limitare il numero di attributi sono i seguenti:

- Gli attributi, a differenza degli elementi, non possono contenere valori multipli
- Gli attributi non sono facilmente espandibili per cambiamenti futuri
- Gli attributi non possono essere usati per descrivere strutture di dati, mentre gli elementi si
- Gli attributi sono più difficili da manipolare da parte del codice di programma
- Gli attributi rendono i documenti più difficili da leggere e da mantenere

Un eccezione a quanto detto è la seguente: si può utilizzare un attributo, ogni qual volta sono presenti, all'interno di un documento XML, più elementi dello stesso tipo, per individuare in maniera univoca un determinato elemento, come ad esempio un attributo ID:

```
<messaggi>
  <appunto id="n1">
```

```
    <da>Basilio</da>
    <a>Ercole</a>
    <oggetto>Conferma</oggetto>
    <body>Appuntamento confermato per le ore 16:00.</body>
  </appunto>

  <appunto id="n2">
    <da>Ercole</da>
    <a>Basilio</a>
    <oggetto>Conferma</oggetto>
    <body>Non posso essere presente per le ore 16:00. Rinviamo di
          un'ora.</body>
  </appunto>
</messaggi>
```

6.7 SPAZIO DEI NOMI IN XML

Il linguaggio XML, come precedentemente visto, non ha dei nomi di elementi prefissati, ma essi vengono creati dall'autore di un documento.
Quando in uno stesso documento vengono a trovarsi elementi diversi, ma con lo stesso nome, possono verificarsi dei conflitti.

Una prima possibile soluzione può essere quella di usare un prefisso diverso da associare ad ogni elemento. Come per esempio, nel caso dei due elementi: "prìncipi" e "princìpi", si possono usare i prefissi "a" e "b":

```
<a:principi>Carlo di Inghilterra</a:principi>
<b:principi>Relatività</b:principi>
```

In tal modo, il conflitto è stato risolto poiché vengono usati due nomi differenti.
Un metodo migliore per la risoluzione dei nomi degli elementi è quello di utilizzare uno *spazio dei nomi* (*namespace*).

In tal caso, piuttosto che usare solo dei prefissi, viene specificato nel tag iniziale dell'elemento, un attributo *xmlns* per assegnare al prefisso uno spazio dei nomi ben preciso, la cui sintassi è la seguente:

```
<prefisso:nomeElemento xmlns:prefisso="spazioDeiNomi">
```

come ad esempio:

```
<a:principi xmlns:a="http://xXx.org/namespace">
```

Le specifiche del W3C affermano che il namespace dovrebbe essere un URI (Uniform Resource Identifier) [B1].

Quando un prefisso è associato ad un elemento, automaticamente tutti gli elementi figli con lo stesso prefisso vengono associati allo stesso namespace.
In realtà, l'indirizzo utilizzato per identificare lo spazio dei nomi non viene usato dal parser per cercare alcuna informazione. L'unico scopo è quello di dare al namespace un nome univoco. Comunque, molto spesso lo spazio dei nomi viene usato come link ad una pagina web contenente informazioni sullo spazio dei nomi stesso.

Per evitare di usare i prefissi per tutti gli elementi figli del tag principale, si utilizza il cosiddetto *namespace di default* avente la seguente sintassi:

```
<nomeElemento xmlns="spazioDeiNomi">
```

come ad esempio:

```
<principi xmlns="http://xXx.org/namespace">
  <nome>Carlo</nome>
```

```
  <luogo>Inghilterra</luogo>
</principi>
```

6.8 XML SCHEMA

Uno schema XML è un metodo per descrivere la struttura, cioè i blocchi legali, di un documento XML permettendo la validazione del documento stesso.

Uno schema XML definisce:

- Gli elementi che possono apparire in un documento
- Gli attributi che possono apparire in un documento
- Quali elementi sono elementi figli
- L'ordine degli elementi figli
- Il numero di elementi figli
- Se un elemento è vuoto o può includere testo
- Tipi di dati per elementi ed attributi
- Valori fissati e di default per elementi ed attributi

Dato il seguente documento XML chiamato "appunto.xml":

```
<?xml version="1.0"?>
<appunto xmlns="http://www.w3schools.com"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3schools.com
                             http://xXx.org/appunto.xsd">
  <da>Basilio</da>
  <a>Ercole</a>
  <oggetto>Conferma</oggetto>
```

```
  <body>Appuntamento confermato per le ore 16:00.</body>
</appunto>
```

il corrispondente schema XML, "appunto.xsd", che definisce gli elementi del documento sopra, è il seguente:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.w3schools.com"
           xmlns="http://www.w3schools.com"
           elementFormDefault="qualified">
<xs:element name="appunto">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="da" type="xs:string"/>
      <xs:element name="a" type="xs:string"/>
      <xs:element name="oggetto" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Da notare che in questo esempio si utilizza il prefisso **xs**.
Ogni schema XML deve avere un elemento **<schema>** che costituisce *l'elemento radice*.
Esso può anche contenere alcuni attributi.

L'insieme di attributi dell'elemento schema costituisce la *dichiarazione di schema*:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.w3schools.com"
           xmlns="http://www.w3schools.com"
```

```
elementFormDefault="qualified">
```

Il primo attributo:

```
xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

sta ad indicare la provenienza degli elementi (schema, element, complexType, sequenze, ecc.) e dei tipi di dati (string, boolean, ecc.) "usati" nello schema, cioè lo spazio dei nomi che in questo caso risulta essere: **"http://www.w3.org/2001/XMLSchema"**. L'attributo indica anche che tali elementi e tipi di dati dovrebbero essere prefissati da "xs".

La linea di codice seguente:

```
targetNamespace="http://www.w3schools.com"
```

indica che gli elementi "definiti" nello schema (appunto, da, a, oggetto e body) provengono dallo spazio dei nomi **"http://www.w3schools.com"**.

```
xmlns="http://www.w3schools.com"
```

indica che lo spazio dei nomi di default è: **"http://www.w3schools.com"**.

Infine, il frammento di codice:

```
elementFormDefault="qualified"
```

indica che gli elementi usati nel documento XML, dichiarati in questo schema, devono essere qualificati dal namespace.

Per quanto riguarda il documento XML, appunto.xml, esso contiene un

riferimento al relativo schema XML tramite gli attributi dell'elemento radice.

Il primo attributo:

```
xmlns="http://www.w3schools.com"
```

è la dichiarazione dello spazio dei nomi di default. Esso dice al validatore di schema che tutti gli elementi usati nel documento sono dichiarati nello spazio dei nomi **"http://www.w3schools.com"**.

La seconda linea di codice:

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

indica lo spazio dei nomi dell'istanza dello schema XML.

Infine l'attributo **schemaLocation**:

```
xsi:schemaLocation="http://www.w3schools.com
                    http://xXx.org/appunto.xsd">
```

indica una coppia di valori che hanno rispettivamente il significato di namespace da usare (**http://www.w3schools.com**) e la posizione dello schema XML per la validazione del documento (**http://xXx.org/appunto.xsd**).

6.8.1 Definizione di elementi in uno schema XML

Gli elementi XML possono essere *semplici* e *complessi*.

Si definisce elemento semplice un elemento XML che non può contenere altri elementi o attributi.

Un elemento semplice perciò può contenere solo *testo* ovvero i tipi di dati inclusi nella definizione dello schema XML (boolean, string, date, ecc.) o un tipo definito dall'utente.

La sintassi di definizione di un elemento semplice all'interno di uno schema XML è la seguente:

```
<xs:element name="xxx" type="yyy"/>
```

dove **xxx** è il nome e **yyy** il tipo di dato dell'elemento.

Dove gli elementi: da, a, oggetto e body sono così definiti:

```
<xs:element name="da" type="xs:string"/>
<xs:element name="a" type="xs:string"/>
<xs:element name="oggetto" type="xs:string"/>
<xs:element name="body" type="xs:string"/>
```

I tipi di dati definiti nello spazio dei nomi dell'XML schema sono i seguenti:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Si definisce elemento complesso un elemento XML che contiene altri elementi e/o attributi.

Ci sono quattro tipi di elementi complessi, che sono:

- *elementi vuoti* – elementi che possono contenere solo attributi, ma che non possono avere né elementi nè altro contenuto tra i tag di apertura e chiusura.
- *Elementi che contengono solo altri elementi* – elementi che possono contenere altri elementi e attributi, ma che non possono avere altro contenuto.
- *Elementi che contengono solo testo* – elementi che possono contenere attributi e testo, ma non elementi.
- *Elementi che contengono sia altri elementi che testo* – elementi generici che possono contenere sia attributi, che elementi, che testo.

Un elemento complesso può essere definito in diversi modi:

1. l'elemento può essere dichiarato direttamente, come nell'esempio:

```
<xs:element name="appunto">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="da" type="xs:string"/>
      <xs:element name="a" type="xs:string"/>
      <xs:element name="oggetto" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

In tal modo solo l'elemento **appunto** può usare il tipo complesso specificato. Dato che gli elementi figli sono racchiusi dall'indicatore **<sequence>**, essi

devono apparire nello stesso ordine in cui compaiono nello schema.

2. l'elemento può avere un attributo type che riferisce il nome del tipo complesso da usare:

```
<xs:element name="appunto" type=”info”>
  <xs:complexType name=”info”>
    <xs:sequence>
      <xs:element name="da" type="xs:string"/>
      <xs:element name="a" type="xs:string"/>
      <xs:element name="oggetto" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

In tal modo, diversi elementi possono riferirsi al tipo complesso, specificando nell'attributo type il valore **info**.

6.8.2 Definizione di attributi in uno schema XML

Tutti gli attributi sono dichiarati come tipi semplici e,come visto sopra, solo gli elementi complessi possono avere attributi.

La sintassi di definizione di un attributo è la seguente:

```
<xs:attribute name="xxx" type="yyy"/>
```

dove **xxx** è il nome e **yyy** il tipo di dato dell'attributo.
Un esempio di utilizzo è il seguente:

```
<nome lingua="IT">Mario</nome>
```

mentre la corrispondente definizione di tale attributo è:

```
<xs:attribute name="lingua" type="xs:string"/>
```

I tipi di dati per gli attributi definiti nello spazio dei nomi dell'XML schema sono i seguenti:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

6.9 FORMATO DI RISPOSTA XML

Il protocollo OAI-PMH utilizza XML come linguaggio di rappresentazione delle risposte inviate a seguito delle richieste effettuate da parte dell'harvester. Tutte le risposte ritornate devono avere i seguenti requisiti:

- essere documenti XML ben formati
- usare la codifica di carattere UTF-8 di rappresentazione Unicode
- devono essere validate attraverso lo schema XML dell'OAI-PMH v.2.0 che trovasi all'indirizzo:
  http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd

Tutte le risposte OAI-PMH devono anche contenere degli elementi comuni:

- il primo tag rappresenta la dichiarazione XML del tipo:
  <?xml version="1.0" encoding="UTF-8"?>
  dove **version** rappresenta la versione del linguaggio, mentre **encoding** la codifica di rappresentazione del carattere adottata
- l'elemento radice con il nome OAI-PMH che racchiude tutto il contenuto della risposta. Esso deve avere tre attributi di definizione dello spazio dei nomi XML usato nel resto della risposta e la locazione dello schema XML di validazione. Essi sono:
  - *xmlns* – il cui valore rappresenta l'URI dello spazio dei nomi OAI-PMH che è http://www.openarchives.org/OAI/2.0/
  - *xmlns:xsi* – il cui valore rappresenta l'URI dello spazio dei nomi per lo schema XML che è:
    http://www.w3.org/2001/XMLSchema-instance
  - *xsi:schemaLocation* – il cui valore corrisponde ad una coppia in cui la prima parte rappresenta l'URI dello spazio dei nomi OAI-PMH, cioè http://www.openarchives.org/OAI/2.0/, mentre la seconda è l'URL dello schema XML di validazione della risposta, cioè l'indirizzo
    http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd
- i primi due elementi figli dopo la radice che sono:
  - *responseDate* – che indica la data e l'ora, da esprimere in formato UTC, in cui la risposta è stata inoltrata
  - *request* – che indica la richiesta del protocollo che ha generato la risposta. Le regole per generare l'elemento *request* sono le seguenti:

    Il contenuto dell'elemento *request* deve sempre essere l'URL

dell'harvester che ha generato la richiesta

Se non avvengono errori o condizioni di eccezione, gli attributi e i valori dell'elemento *request* devono corrispondere agli argomenti e ai rispettivi valori della richiesta inviata, espressi come coppie nome-valore

Se avvengono errori o condizioni di eccezione, l'elemento *request* dovrà contenere all'interno l'URL dell'harvester che ha generato la richiesta e non dovrà specificare alcun attributo.

- Il terzo elemento figlio della radice è uno dei due seguenti:
  - Un elemento *error* che deve essere usato nel caso in cui si verifichi un errore o una condizione d'eccezione
  - Un elemento che ha lo stesso nome del verbo della richiesta OAI-PMH

Un esempio di risposta alla richiesta *GetRecord* [B1] in cui non si verificano né errori né condizioni di eccezione è il seguente:

```
<?xml version="1.0" encoding="UTF-8" ?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
                 http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
  <responseDate>2002-05-01T19:20:30Z</responseDate>
  <request verb="GetRecord"
           identifier="oai:arXiv.org:hep-th/9901001"
           metadataPrefix="oai_dc">http://an.oa.org/OAI-script
  </request>
  <GetRecord>
    <record>...</record>
  </GetRecord>
```

```
</OAI-PMH>
```

Un esempio di risposta alla richiesta *GetRecord* in cui si verifica un errore di tipo *idDoesNotExist* è il seguente:

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
                 http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
  <responseDate>2002-02-08T08:55:46Z</responseDate>
  <request verb="GetRecord"
           identifier="oai:arXiv.org:quant-ph/0213001"
           metadataPrefix="oai_dc">http://arXiv.org/oai2
  </request>
  <error code="idDoesNotExist">No matching identifier in arXiv
  </error>
</OAI-PMH>
```

Lo schema XML per la validazione delle risposte, utilizzato dall'OAI-PMH versione 2.0 può essere reperito al seguente indirizzo:

http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd

BIBLIOGRAFIA

[B1] Amelotti Ercole, *Protocollo OAI-PMH negli Open Archive e applicazione CDSware per la rappresentazione dei relativi dati bibliografici*, tesi di laurea in informatica, Università degli Studi di Messina, A.A. 2003-2004 (relatore Puccio L., correlatore De Robbio A.).

SITOGRAFIA

[S13] http://www.w3schools.com/xml/default.asp

[S14] http://www.openarchives.org/OAI/openarchivesprotocol.html