UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

XML
Extensible Markup Language

UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

XML – Extensible Markup Language (1/3)

Extensible Markup Language (XML) è un formato testuale derivante da SGML (ISO 8879):

- XML è stato progettato per descrivere dati
- XML è un tool indipendente dal software e dall'hardware per trasportare informazioni
- XML è uno standard W3C (World Wide Web Consortium)


```
<?xml version="1.0" encoding="UTF-8"?>
<foo>Hello World!</foo>
```

UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

XML – Extensible Markup Language (2/3)

XML non è:

- Un rifacimento di HTML
  (ma HTML può essere generato da XML)
- Un formato di presentazione
  (ma XML può essere convertito in un formato di presentazione)
- Un linguaggio di programmazione
- Un protocollo di rete
- Un database


```
<?xml version="1.0" encoding="UTF-8"?>
<foo>Hello World!</foo>
```

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

XML – Extensible Markup Language (3/3)

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

Prologo

Il prologo di un documento XML può contenere alcuni dei seguenti campi di seguito riportati, che forniscono informazioni rigurado il documento XML in esame:

Dichiarazione XML: che riporta informazioni sul format XML del document.

Es: <?xml version="1.0" encoding="UTF-8"?>

Commenti

Es: <!- - This is a comment - - >

Processing instructions (PIs): istruzioni dirette al processore XML

Es: <?xml-stylesheet type="text/css" href="Example.css"?>

DTD "inline"

Es: <!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>]>

External DTD

Es: <!DOCTYPE slideshow SYSTEM "Example.dtd">

Dichiarazione XML

Nella dichiarazione XML è possibile riscontrare i seguenti campi

- **Version:** identifica la versione di XML in uso.
- **Encoding:** specifica quale tipo di codifica deve essere adoperata all'interno del documento
- **Standalone**: indica la presenza di eventuali riferimenti a documenti esterni.

Esempio

<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>

Processing Instruction

Processing instruction non sono una componente della parte dati del documento XML ma sono delle direttive dirette al processore XML:

Le Processing instruction sono comprese tra <? e ?>

Esempio

<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>

Tag

I tag XML non ha un significato pre-definito. I tag XML sono case-sensitive.

```
<Message>This is incorrect</message>
<message>This is correct</message>
```

Nella definizione dei tag non possono essere usati:

- Spazi
- Caratteri speciali (virgolette, percentuale, punto e virgola…)

Tra un tag di apertura e un tag di chiusura possono essere inseriti:

- Altri tag
- Dati testuali
- Contenuto misto

UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

Attributi

Un attributo è una coppia (nome,valore) associata al tag di partenza di un elemento. Il valore è racchiuso tra doppi o singoli apici. Un elemento può avere uno o più attributi

```
<Libro Titolo="Basi di dati per la gestione dell'informazione">
   <Nome>Angelo</Nome>
   <Cognome>Chianese</Cognome>
   <Nome>Antonio</Nome>
   <Cognome>Picariello</Cognome>
   <Nome>Vincenzo</Nome>
   <Cognome>Moscato</Cognome>
   <Nome>Lucio</Nome>
   <Cognome>Sansone</Cognome>
   <Editore>McGraw-Hill</Editore>
</Libro>
```

UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

XML-
Well formed

"A data object is an XML document if it is well-formed, as defined in the XML 1.0 specification. A well-formed XML document may in addition be valid if it meets certain further constraints"

Affinché un documento XML sia ben formato deve rispettare alcune regole sintattiche:

1. Ad ogni tag iniziale deve corrispondere un tag finale

2. Gli elementi non possono sovrapporsi

3. Deve esistere uno ed un solo elemento radice

4. I valori degli attributi devono essere specificati tra apici

5. I commenti e le istruzioni di elaborazione non possono apparire all'interno dei tag

6. Non deve apparire nessun carattere < o & all'interno dei caratteri di un elemento o di un attributo

Riferimenti ad entità

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

Sezioni CDATA

Una sezione CDATA può essere adoperata per gestire grandi moli di testo, in quanto al suo interno è possibile adoperare un qualunque carattere.

Una sezione CDATA inizia con la sequenza <![CDATA[e termina con la sequenza]]>

```
<script>
<![CDATA[
function matchwo(a,b){
if (a < b && a < 0) then{
  return 1;}
else{
  return 0;}}]]>
</script>
```

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

XML Namespaces

I namespace XML permettono di evitare i conflitti di nomi mediante l'utilizzo di prefissi. Il namespace è definito dall'attributo **xmlns** nel tag iniziale di un elemento secondo la seguente sintssi xmlns:*prefix*="*URI*".

```
<root>
<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
<f:table xmlns:f="http://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
</root>
```

UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

DTD – Document Type Definition

Un documento XML è ben formato se è conforme alle regole definite nel DTD (Document Type Definition). L'obiettivo di un DTD, che può essere inserito in un file esterno o nello stesso documento XML, è di definire la struttura del documento XML.

La sintassi è la seguente:

<!ELEMENT nome_elemento (modello_di_contenuto) >

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

DTD – Example

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this
weekend!</body>
</note>
```

DTD

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

DTD – Elementi(1/2)

Le regole permettono di definire per ciascun elemento la tipologia dell'elemento, il numero di figli ed il loro ordine:

- #PCDATA: Specifica che l'elemento può contenere solamente dati di tipo carattere.
  <!ELEMENT cognome (#PCDATA)> → <cognome>Picariello</cognome>

- Elementi figli: Specifica il tipo di figli che un elemento deve avere. E' possibile aggiungere un suffisso per specificare il numero di istanze di figli permessi per un dato elemento
  - ? corrisponde a "zero o un elemento"
  - * corrisponde a "zero o più elementi"
  - + corrisponde a "uno o più elementi"

  <!ELEMENT fax (numero_di_telefono)> →
  <fax><numero_di_telefono>0000012</numero_di_telefono></fax>

UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

DTD – Elementi(2/2)

- Sequenze: Una sequenza permette di definire l'ordine con cui far apparire gli elementi.
  <!ELEMENT persona (Nome, Cognome, Data_di_nascita)>

- Scelte: un elenco di nomi separati da barre verticali |. E' possibile concatenare le scelte effettute.
  <!ELEMENT cerchio (colore, (raggio | diametro))>

- Elementi vuoti: vengono dichiarati usando la parola EMPTY
  <!ELEMENT sposato EMPTY>

- ANY: La parola ANY consente di specifiare un contenuto arbitrario.
  <!ELEMENT page ANY>

- Contenuto misto: un certo elemento può avere contenuto formato da due elementi definiti in precedenza .
  <!ELEMENT page (#PCDATA | elemento_figlio)* >

UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

DTD – Attributi

La dichiarazione degli attributi ha la seguente forma:

<!ATTLIST elemento attributo caratteristiche >

- **Elemento** è il nome dell'elemento che deve contenere l'attributo
- **Attributo** è il nome dell'attributo
- **Caratteristiche** permettono di specificare
  - il tipo di attributo
  - se l'attributo è obbligatorio oppure opzionale

Ogni dichiarazione ATTLIST può includere un valore di presenza dell'attributo

- **#IMPLIED**: Indica che l'attributo è opzionale
- **#REQUIRED**: Indica che l'attributo è obbligatorio
- **#FIXED**: Indica che l'attributo è costante e non può essere variato

<!ATTLIST persona nome CDATA #REQUIRED cognome CDATA #REQUIRED eta CDATA #IMPLIED nazionalita CDATA #FIXED "Italiana">

UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

XML - Schema (1/2)

Uno XML Schema descrive la struttura di un documento XML. Un XML Schema definisce:

- gli elementi e gli attributi che possono apparire in un documento
- per ciascun elemento, quali sono gli elementi figli ed in quale ordine devono apparire
- se un elemento deve essere vuoto o può contenere testo
- il tipo di ciascun elemento e ciascun attributo
- il valore di default per tutti gli elementi ed attributi

In XML Schema:

- Gli elementi che includono altri elementi o che hanno attributi sono di tipo **complex type**. Gli elementi di tipo **Complex type** devono essere esplicitamente definiti dall'utente.
- Gli Elementi che contengono solo dati (es. numeri, stringhe, date, etc) e non hanno sotto elementi sono detti di tipo **simple type.**

XML - Schema (2/2)

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

XSLT

EXtensible Stylesheet Language Transformations (XSLT) è un linguaggio che permette di trasformare documenti XML in altri formati.

XSL permette di manipolare e formattare documenti XML ed è costituito dalle seguenti 3 parti

- XSL – linguaggio per trasformare documenti XML
- XPath – linguaggio per "navigare" documenti XML
- XSL-FO – linguaggio per formattare documenti XML

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

XPath

Xpath è utilizzato per navigare attraverso elementi ed attribute di un documento XML

- XPath è una sintassi per la definizione di parti di un documento XML
- XPath utilizza percorsi per navigare in un documento XML
- XPath contiene una libreria di funzioni standard

Expression	Description
nodename	Selects all nodes with the name "*nodename*"
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

Foglio di Stile

Un insieme di regole che permettono di trasformare un documento in un altro documento si chiama foglio di stile

Il foglio di stile permette di definire un insieme di regole per la trasformare il documento XML in un altro formato.
La radice del documento di stile deve avere la seguente sintassi:

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

I template sono utilizzati per gestire l'output del processo di trasformazione e sono rappresentati da un elemento **xsl:template**, dotato di un attributo match:

- xsl:template definisce una regola di trasformazione
- match(espressione Xpath) identifica il tipo di input che attiva la regola

Foglio di Stile - Esempio (1/3)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0“
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

Foglio di Stile - Esempio (2/3)

L'elemento **xsl:value-of** permette di selezionare il contenuto di un elemento di input e inserirlo come output e si può trovare soltanto all'interno di un elemento **xsl:template**
L'attributo **select** permette di selezionare il valore che deve essere prelevato.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>
<xsl:template match="title">
  Title: <span style="color:#ff0000">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>
</xsl:stylesheet>
```

Foglio di Stile - Esempio (3/3)

L'utilizzo della regola **xsl:applytemplates** permette di stabilire il tipo di template da usare e l'ordine con cui utilizzarli

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>
<xsl:template match="cd">
  <p>
  <xsl:apply-templates select="title"/>
  <xsl:apply-templates select="artist"/>
  </p>
</xsl:template>
</xsl:stylesheet>
```

Foglio di Stile - Esempio (3/3)

L'utilizzo della regola **xsl:applytemplates** permette di stabilire il tipo di template da usare e l'ordine con cui utilizzarli

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>
<xsl:template match="cd">
  <p>
  <xsl:apply-templates select="title"/>
  <xsl:apply-templates select="artist"/>
  </p>
</xsl:template>
</xsl:stylesheet>
```

API

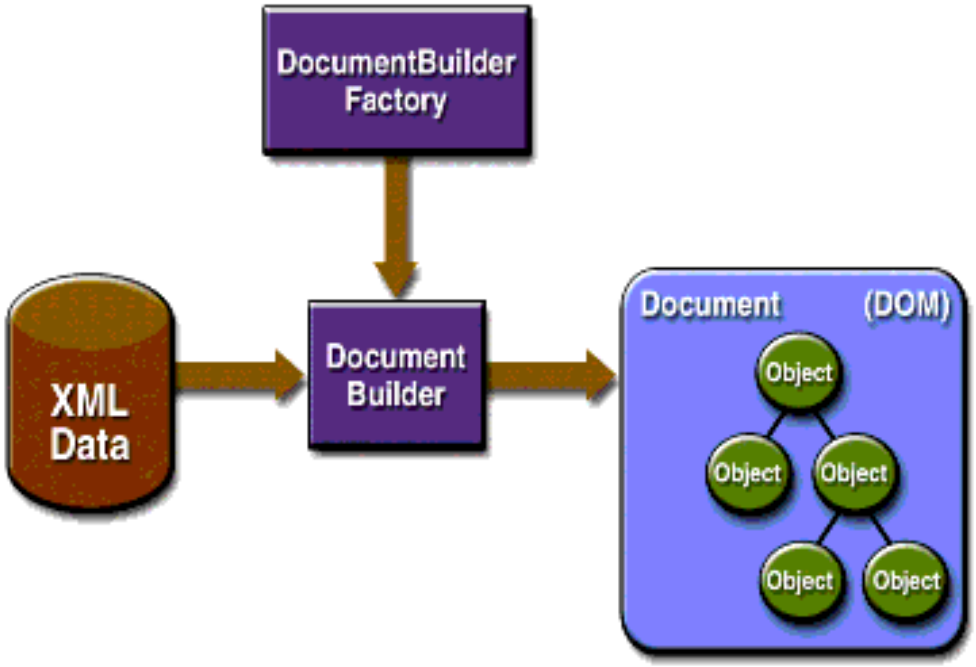
La struttura gerarchica dei documenti XML ha portato alla proliferazione di molteplici librerie per la manipolazione di documenti XML.

Per poter accedere e manipolare i documenti è possibile adoperare due API

1. Document Object Model (DOM)
2. Simple API for XML (SAX)

DOM

Il parser DOM è un'API basata su albero. Un'API basata su un albero fornisce interfacce per le component dell'albero .
Un parser DOM crea una struttura ad albero in memoria dal documento di input. Un parser DOM serve sempre l'applicazione cliente con l'intero documento.



UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

DOM – Example

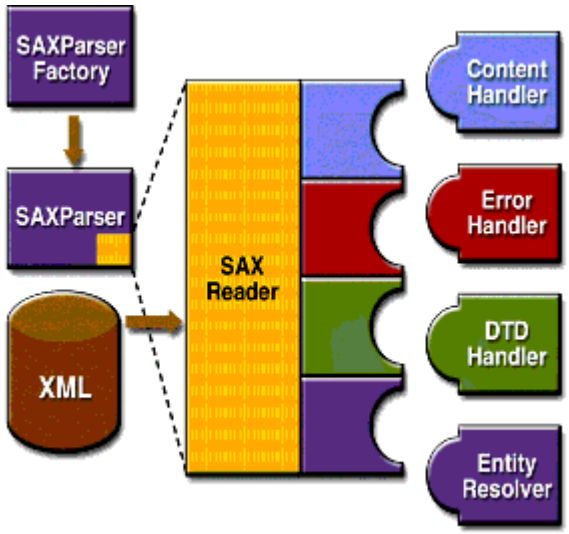
```
<?xml version="1.0"?>
<company> <staff id="1001">
<firstname>yong</firstname>
<lastname>mook kim</lastname>
<nickname>mkyong</nickname>
<salary>100000</salary> </staff>
<staff id="2001">
<firstname>low</firstname>
<lastname>yin fong</lastname>
<nickname>fong fong</nickname>
<salary>200000</salary> </staff>
</company>
```

```
File fXmlFile = new File("/Users/mkyong/staff.xml");
DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document doc = dBuilder.parse(fXmlFile);
doc.getDocumentElement().normalize();
 System.out.println("Root element :" + doc.getDocumentElement().getNodeName());
 NodeList nList = doc.getElementsByTagName("staff");
 for (int temp = 0; temp < nList.getLength(); temp++) {
Node nNode = nList.item(temp);
System.out.println("\nCurrent Element :" + nNode.getNodeName());
if (nNode.getNodeType() == Node.ELEMENT_NODE) {
 Element eElement = (Element) nNode;
System.out.println("Staff id : " + eElement.getAttribute("id"));
System.out.println("First Name : " + eElement.getElementsByTagName("firstname").item(0).getTextContent());
System.out.println("Last Name : " + eElement.getElementsByTagName("lastname").item(0).getTextContent());
System.out.println("Nick Name : " + eElement.getElementsByTagName("nickname").item(0).getTextContent());
System.out.println("Salary : " + eElement.getElementsByTagName("salary").item(0).getTextContent());   } }
```

SAX

SAX è basata sul concetto di evento associato al processo di parsing di un documento. Per adoperare SAX è necessario indicare le azioni da compiere al verificarsi di un determinato evento. Per maneggiare un evento è possibile adoperare uno dei 4 seguenti handler:

- ContentHandler interface
- DTDHandler interface
- EntityResolver interface
- ErrorHandler interface.



UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

SAX – Example

```
<?xml version="1.0"?>
<company> <staff id="1001">
<firstname>yong</firstname>
<lastname>mook kim</lastname>
<nickname>mkyong</nickname>
<salary>100000</salary> </staff>
<staff id="2001">
<firstname>low</firstname>
<lastname>yin fong</lastname>
<nickname>fong fong</nickname>
<salary>200000</salary> </staff>
</company>
```

```
SAXParserFactory factory = SAXParserFactory.newInstance();
SAXParser saxParser = factory.newSAXParser();
DefaultHandler handler = new DefaultHandler() {
boolean bfname=blname=bnname=bsalary= false;
public void startElement(String uri, String localName,String qName, Attributes attributes) throws SAXException {
System.out.println("Start Element :" + qName);
if (qName.equalsIgnoreCase("FIRSTNAME")) { bfname = true; }
if (qName.equalsIgnoreCase("LASTNAME")) { blname = true; }
if (qName.equalsIgnoreCase("NICKNAME")) { bnname = true; }
if (qName.equalsIgnoreCase("SALARY")) { bsalary = true; }   }
public void endElement(String uri, String localName, String qName) throws SAXException {
System.out.println("End Element :" + qName);   }
public void characters(char ch[], int start, int length) throws SAXException {
if (bfname) { System.out.println("First Name : " + new String(ch, start, length)); bfname = false; }
if (blname) { System.out.println("Last Name : " + new String(ch, start, length)); blname = false; }
if (bnname) { System.out.println("Nick Name : " + new String(ch, start, length)); bnname = false; }
if (bsalary) { System.out.println("Salary : " + new String(ch, start, length)); bsalary = false; }   }   };
saxParser.parse("c:\\file.xml", handler);
```

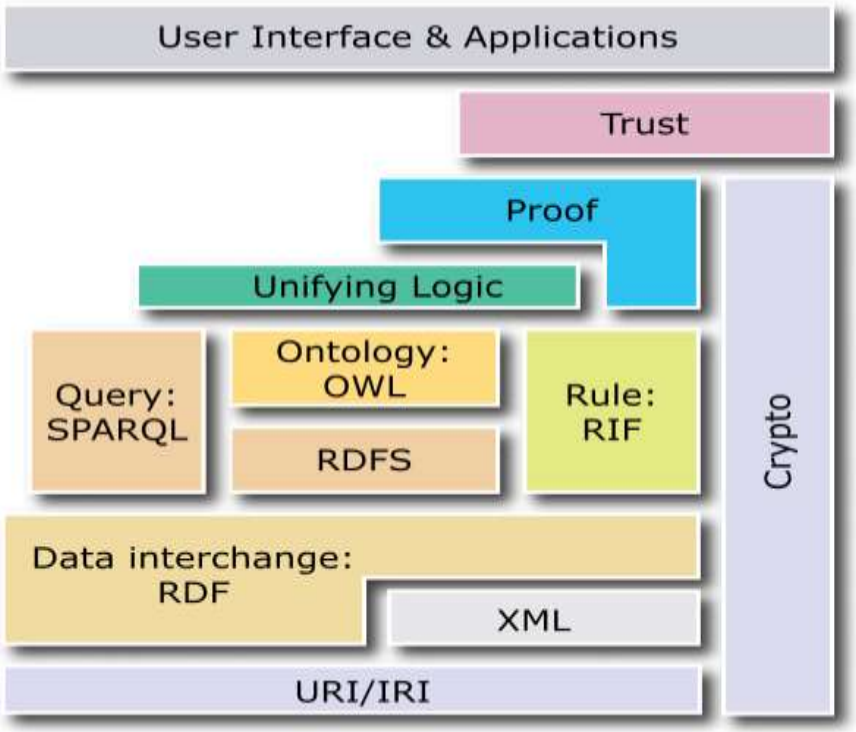
DOM vs SAX

DOM	SAX
Struttura ad albero	Event-Based
Maggiore occupazione di memoria	Minore occupazione di memoria
Manipolazione arbitraria del documento	Solo accesso sequenziale

RDF(1/2)

XML è un meta-linguaggio per definire mark-up. Esso fornisce un formato di interscambio di dati e metadata tra applicazioni. XML non attribuisce significato ai dati.

Resource Description Framework (RDF) è a modello dati per rappresentare le informazioni inerenti le risorse Web. RDF può quindi essere usato per rappresentare risorse web anche se esse non possono essere direttamente recuperate dal web (ad esempio libri, persone …)



RDF(2/2)

I concetti principali di RDF sono:

- Risorse: che rappresentano l'oggetto di interesse e che è rappresentata da un URI. L'URI può essere un'URL o un qualsiasi altro identificativo unico.
- Proprietà: descrivono relazioni tra le risorse e sono identificate tramite URI.
- Statements: sono triple oggetto-attributo-valore. Il valore può essere una risorsa o un letterale.