Introduction to XML

XML: basic elements

XML

"Trying to wrap your brain around XML is sort of like trying to put an octopus in a bottle. Every time you think you have it under control, a new tentacle shows up. XML has many tentacles, reaching out in all directions. "

(Dick Baldwin)

```
<book>
    <chap>
        Text for Chapter 1
    </chap>
    <chap>
        Text for Chapter 2
    </chap>
</book>
```

What is XML?

eXtensible Markup Language, or XML for short, is a new technology for web applications.

XML is a World Wide Web Consortium standard that lets you create your own tags.

XML is not a single technology, but a group of related technologies that continually adds new members

XML is a *lingua-franca* that simplifies business-to-business transactions on the web.

Vendor independence in the data-formatting context

"Other successful Internet technologies let people run their systems without having to take into account another company's own computer systems, notably:
TCP/IP for networking,
Java for programming,
Web browsers for content delivery.
XML fills the data formatting piece of the puzzle."

"These technologies do not create dependencies. It means you can build solutions that are completely agnostic about the platforms and software that you use."

Phipps, IBM's chief XML and Java evangelist

XML Jargon

- Computer people are the world's worst at inventing new jargon.
- XML people seem to be the worst of the worst in this regard.

(Dick Baldwin)

XML
DTD
XSL
XSLT

DOM
SAX
JAXP
JDOM

XML Schema
XPath
XLink
XPointer

XQL
XML-RPC
XSP

Related stuff
SGML XHTML CSS

XML applications

Semantic Web
RDF (Resource Description Framework), OWL, Topic Maps

Web Services
SOAP, UDDI, WSDL, XML-RPC

Configuration files

XML roots: SGML

What is SGML

SGML is an ISO standard (ISO 8879:1986) which provides a formal notation for the definition of generalized markup languages. SGML is not a language in itself. Rather, it is a metalanguage that is used to define other languages.

SGML: the three parts

An SGML document is really the combination of three parts. Let's refer to the parts as files (but they don't have to be separate physical files).

One file contains the content of the document (words, pictures, etc.). This is the part that the author wants to expose to the client.

A second file is the DTD that defines the accepted syntax.

A third file is a *stylesheet* that establishes how the content that conforms to the DTD is to be rendered on the output device. This is how the author wants the material to be presented to the client.

HTML versus SGML

HTML implements some of the concepts derived from SGML but in effect the DTD and the Style Sheet are hard-coded into the browser software.

Because each browser manufacturer has some flexibility in implementing the intended style, the same document will sometimes look different when rendered with two different browsers. This is a (wanted) shortcoming of HTML.

Web page designers are constantly faced with the problem of designing workarounds to compensate for the deficiencies in some versions of some browsers being used to view the page.

SGML - HTML

What the world needs now is...

What the Web community needs is an approach where a standard browser is simply a rendering engine that validates a document according to a given DTD and renders it according to a given stylesheet.

A package deal

The combination of the document, the DTD, and the stylesheet would constitute a package delivered by a server to the browser. The author of the document would provide the DTD and the stylesheet in addition to the data to be rendered. Then the author could be more confident that it would be rendered properly, especially for complex data.

SGML – HTML - XML

The two extremes

With HTML, the DTD and the stylesheet are essentially hard-coded into the browser.

With SGML, the processor requires both a DTD and a stylesheet.

XML, the middle ground

With XML, the DTD is optional but the stylesheet (or some processing mechanism that substitutes for a stylesheet) is required.

XML: element, content, and attribute

What is an element?

An element is a sequence of characters that begins with a *start tag* and ends with an *end tag* and includes everything in between.

```
<chap number="1">Text for Chapter 1</chap>
```

What is the content?

The characters in between the tags (rendered in green in this presentation) constitute the CONTENT.

XML: element, content, and attribute

An element may include optional attributes

The *start tag* may contain optional *attributes*. In this example, a single *attribute* provides the number value for the chapter.

```
<chap number="1">Text for Chapter 1</chap>
```

The characters rendered in blue in the above element constitute an attribute.

The term *attribute* is a commonly used term in computer science and usually has about the same meaning, regardless of whether the discussion revolves around XML, Java programming, or database management: Attributes belong to things, or things have attributes.

XML: tree structure

An XML document must have a root tag.

An XML document is an information unit that can be seen in two ways:

As a linear sequence of characters that contain characters data and markup.

As an abstract data structure that is a tree of nodes.

XML: additional elements

An XML document can contain:

Processing Instructions (PI): **<? … ?>**

Comments **<!-- … -->**

When a XML document is analyzed, character data within comments or PIs are ignored.

The content of comments is ignored, the content of PIs is passed on to applications.

XML: CDATA sections

Note: the element content that are going to be parsed are called **PCDATA**

An XML document can contain sections used to escape character strings that may contain elements that you do not want to be examined by your XML engine, e.g. special chars (<) or tags:

CDATA sections **<![CDATA[…]]>**

When a XML document is analyzed, character data within a CDATA section are not parsed, by they remain as part of the element content.

```
<java>
<![CDATA[
    if (arr[indexArr[4] ]>3) System.out.println("<HTML>");
]]>
</java>
```

Avoid having]]> in your CDATA section!

Well formed documents

All XML documents must be well-formed

XML documents need not be valid, but all XML documents must be well-formed.

(HTML documents are not required to be well-formed)

There are several requirements for an XML document to be well-formed.

Well formed documents

Caution: XML is case sensitive

Start and end tags are required

To be well-formed, all elements that can contain character data must have both start and end tags.

(Empty elements have a different requirement: see later.)

For purposes of this explanation, let's just say that the *content* that we discussed earlier comprises character data.

Elements must nest properly

If one element contains another element, the entire second element must be defined inside the start and end tags of the first element.

Well formed documents

Dealing with empty elements

We can deal with empty elements by writing them in either of the following two ways:

```
<book></book>
<book/>
```

You will recognize the first format as simply writing a start tag followed immediately by an end tag with nothing in between.

The second format is preferable

Empty element can contain attributes

Note that an empty element can contain one or more attributes inside the start tag:

```
<book author="eckel" price="$39.95" />
```

Well formed documents

No markup characters are allowed

For a document to be well-formed, it must not have some characters (entities) in the text data: < > " ' &.

If you need for your text to include the < character you can represent it using < or < or &#x3C instead.

All attribute values must be in quotes (apostrophes or double quotes).

You can surround the value with apostrophes (single quotes) if the attribute value contains a double quote. An attribute value that is surrounded by double quotes can contain apostrophes.

Logical structure of an XML document

XML declaration (optional, but if present MUST be the first element)

<?xml version='1.0' encoding='utf-8'>

Optional DTD declaration

Optional comments and Processing Instructions

The root element's start tag

All other elements, comments and PIs

The root element closing tag

XML: namespaces

How do you avoid tag conflicts?

Since you can define your own tags, if you reuse XML files from other authors you might find tag conflicts.

These can be avoided by declaring a namespace as an attribute of the root element:

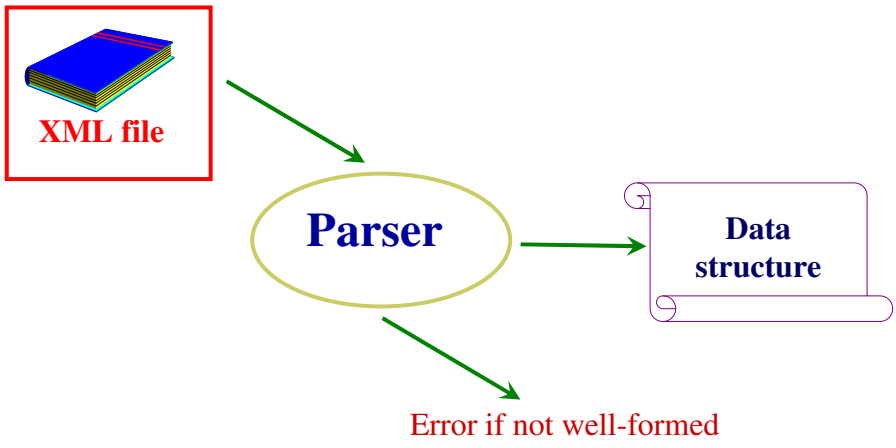
<xsl:stylesheet version ="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

What is a parser?

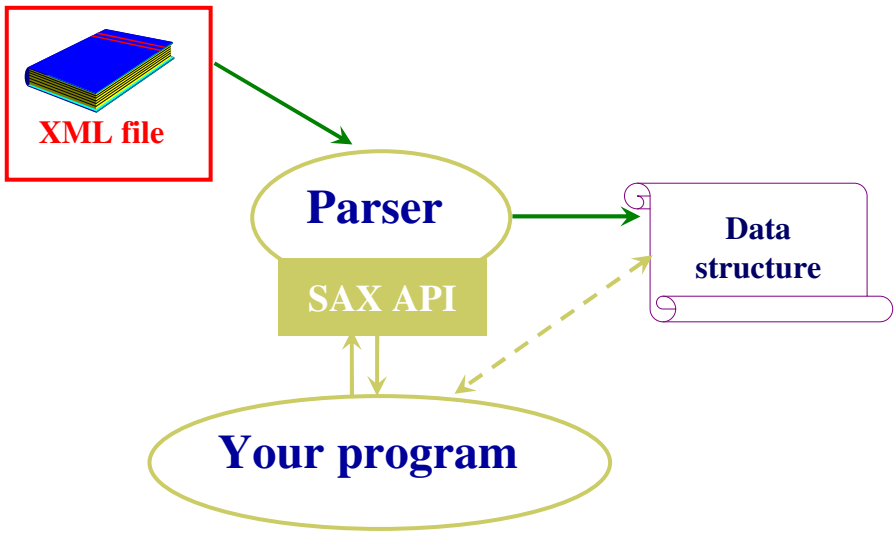
A parser, in this context, is a software tool that preprocesses an XML document in some fashion, handing the results over to an application program.

The primary purpose of the parser is to do most of the hard work up front and to provide the application program with the XML information in a form that is easier to work with.

Making sense of XML: the Parser



Making sense of XML:the Parser



Tree-based vs Event-based API

- **Tree-based API**
  A tree-based API compiles an XML document into an internal tree structure. This makes it possible for an application program to navigate the tree to achieve its objective. The Document Object Model (DOM) working group at the W3C is developing a standard tree-based API for XML.
- **Event-based API**
  An event-based API reports parsing events (such as the start and end of elements) to the application using *callbacks*. The application implements and registers event handlers for the different events. Code in the event handlers is designed to achieve the objective of the application. The process is similar (but not identical) to creating and registering event listeners in the Java Delegation Event Model.

what is SAX?

SAX is a set of interface definitions

For the most part, SAX is a set of interface definitions. They specify one of the ways that application programs can interact with XML documents.

(There are other ways for programs to interact with XML documents as well. Prominent among them is the Document Object Model, or DOM)

SAX is a standard interface for event-based XML parsing, developed collaboratively by the members of the XML-DEV mailing list. SAX 1.0 was released on Monday 11 May 1998, and is free for both commercial and noncommercial use.

The current version is SAX 2.0.1 (released on 29-January 2002)

See http://www.saxproject.org/

Some available Parser

Apache Xerces	http://xml.apache.org
IBM XMLJ4	http://alphaworks.ibm.com/tech/xmlj4
James Clark's XP	http://www.jclark.com/xml/xp
OpenXML	http://www.openxml.org
Oracle XML Parser	http://technet.oracle.com/tech/xml
Sun Microsystem Project X	http://java.sun.com/products/xml
Tim Bray's Lark and Larval	http://www.textuality.com/Lark

Introduction to XML

DTD

What is a DTD?

A DTD is usually a file (or several files to be used together) which contains a formal definition of a particular type of document. This sets out what names can be used for elements, where they may occur, and how they all fit together.

It's a formal language which lets processors automatically parse a document and identify where every element comes and how they relate to each other, so that stylesheets, navigators, browsers, search engines, databases, printing routines, and other applications can be used.

A DTD contain metadata relative to a collection of XML docs.

Valid documents

a *valid* XML document is one that conforms to an existing DTD in every respect.

For example...

Unless the DTD allows an element with the name *"color"*, an XML document containing an element with that name is not valid according to that DTD (but it might be valid according to some other DTD).

An *invalid* XML document can be
a perfectly good and useful XML document.

Valid documents

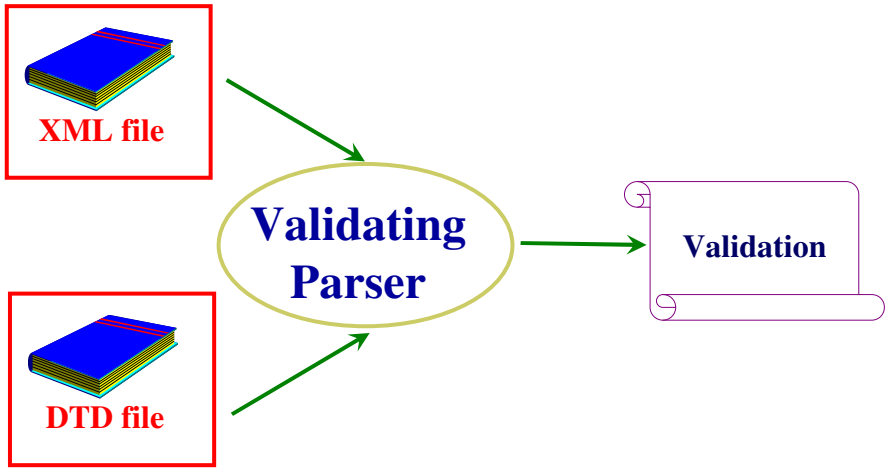
Validity is not a requirement of XML

Because XML does not require a DTD, in general, an XML processor cannot require validation of the document.

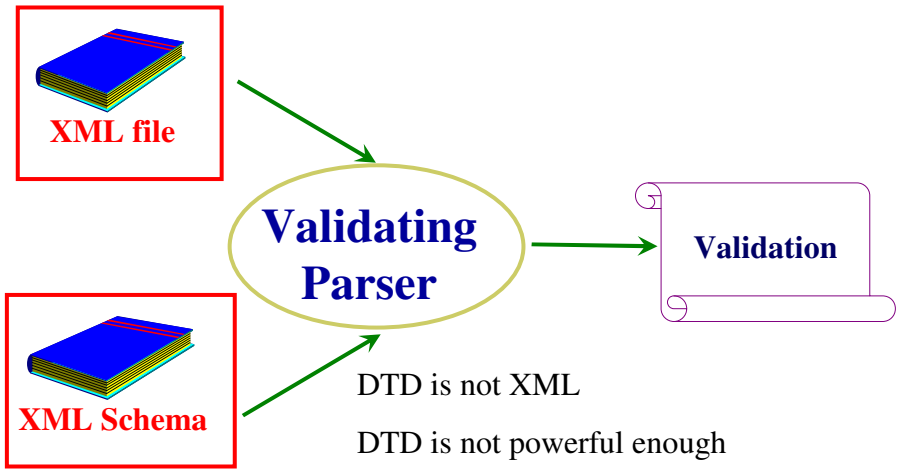
Many very useful XML documents are not valid, simply because they were not constructed according to an existing DTD.

To make a long story short,
validation against a DTD can often be very useful, but is not required.

Constraing &ValidatingXML



Constraing & Validating XML



DTD is not XML

DTD is not powerful enough

(e.g. at least 3, no more than 5)

Where are the DTDs?

A DTD can be external or internal to a document.

```
<!DOCTYPE Report>
<!DOCTYPE Report SYSTEM "Report.dtd">
<!DOCTYPE Report PUBLIC "Report.dtd">
```

Internal DTD

External DTD

Broadly and publicly available

URL

DTD Markup: ELEMENT

```
<!ELEMENT name content-model>
<!ELEMENT book (preface?,chapter+,index)>
<!ELEMENT preface(paragraph+)>
<!ELEMENT paragraph (#PCDATA)>

<!ELEMENT chapter (title,paragraph+,reference*)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT reference (#PCDATA|URL)>
<!ELEMENT URL (#PCDATA)>

<!ELEMENT index(number,title,page_number)>
<!ELEMENT number(#PCDATA)>
<!ELEMENT page_number(#PCDATA)>
```

? Zero or one
+ One or more
* Zero or more
, sequence
| or (not xor!)

DTD Markup: ATTLIST

```
<!ATTLIST element-name attribute-name type default>
<!ELEMENT Product (#PCDATA)>
<!ATTLIST Product
    Name CDATA #IMPLIED
    Rev CDATA #FIXED "1.0"
    Code CDATA #REQUIRED
    Pid ID #REQUIRED
    Series IDREF
    Status (InProduction|Obsolete)
        "InProduction"
>
```

TYPES:
CDATA character data
ID Unique key
IDREF Foreign Key
(…|…) Enumeration

DEFAULT:
#IMPLIED optional, no default
#FIXED optional, default supplied.
If present must match default
#REQUIRED must be provided

DTD Markup: ENTITY

Entities are a sort of macro

General Entity

```
<!ENTITY author "Marco Ronchetti, Universita' di Trento">
```

External Parsed Entity

```
<!ENTITY content SYSTEM "content.xml">
<Tag>&content &author</Tag>
```

External to the DTD

Parameter Entity

```
<!ENTITY % AI "CDATA #IMPLIED">
<!ATTLIST Product Name %AI>
```

Internal at the DTD

The main problem of DTD's...

They are not written in XML!

Solution:

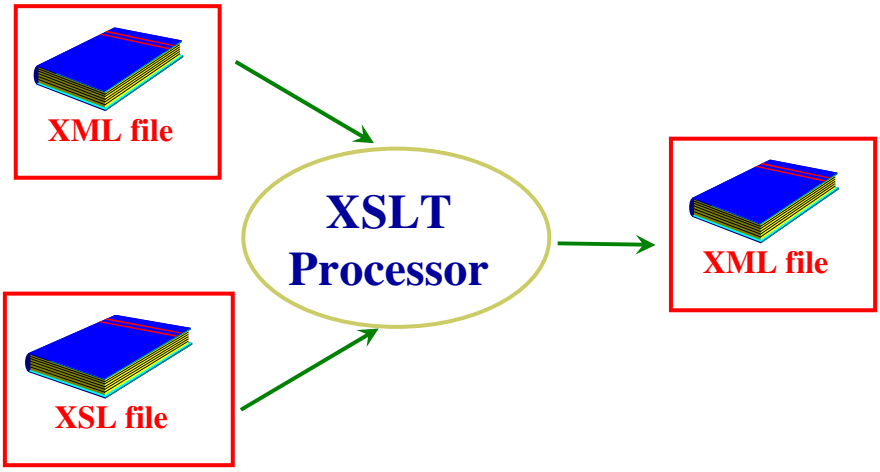
Another XML-based standard: XML Schema

For more info see:
http://www.w3.org/XML/Schema

Introduction to XML

XSL - INTRODUCTION

Transforming XML



XSL is complex

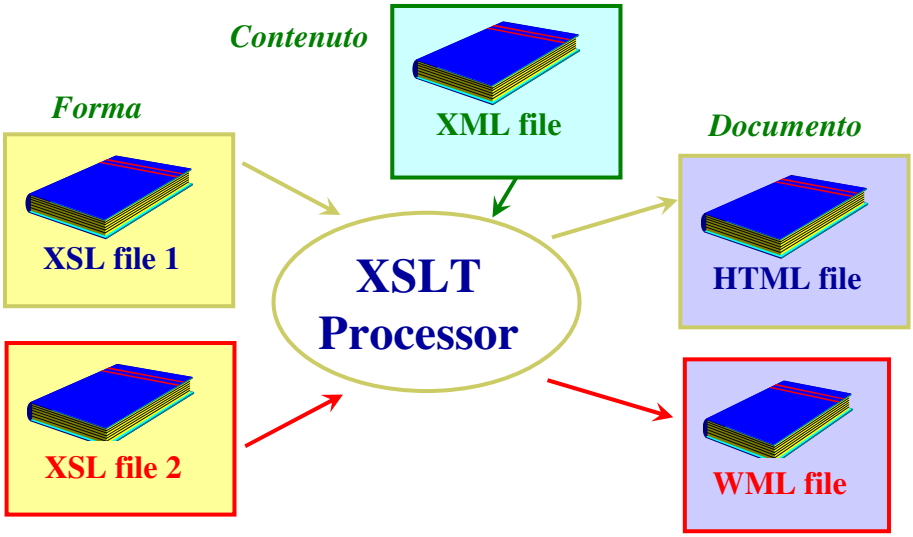
XSL is complex (much more complex than XML). Designing an XSL stylesheet, to be used by a rendering engine to properly render an XML document, can be a daunting task.

Microsoft has developed an XSL debugger, and has made it freely available for downloading.

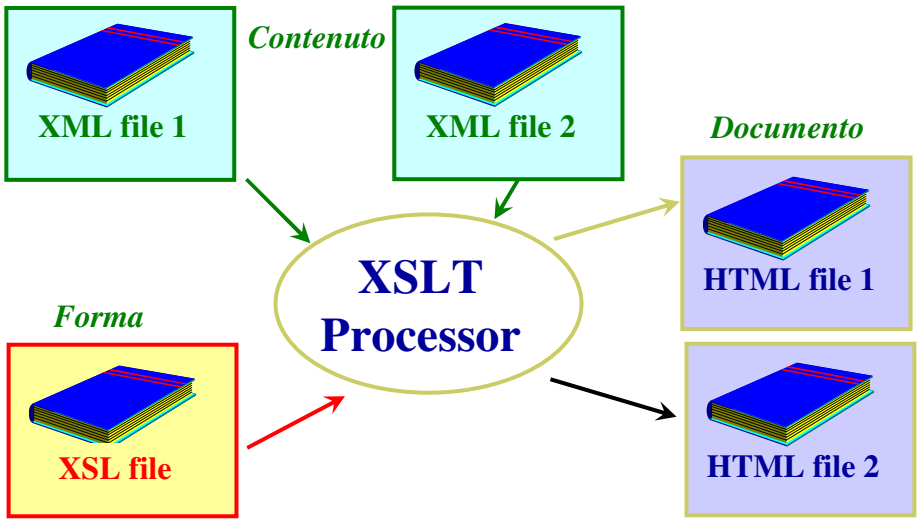
Some availableXSLT processors

Apache Xalan	http://xml.apache.org
James Clark's XT	http://www.jclark.com/xml/xt.html
Lotus XSL Processor	http://alphaworks.ibm.com/tech/LotusXSL
Oracle XSL Processor	http://technet.oracle.com/tech/xml
Keith Visco's XSL:P	http://www.clc-marketing.com/xslp
Michael Kay's SAXON	http://users.iclway.co.uk/mhkay/saxon

Transforming XML



Transforming XML



Hands on XSL

XSL-basic elements

HANDS ON! - Esempio1 XML

```
<?xml version="1.0"?>
<?xml-stylesheet href="hello.xsl" type="text/xsl"?>

<!-- Here is a sample XML file -->
<page>
   <title>Test Page</title>
   <content>
      <paragraph>What you see is what you get!</paragraph>
   </content>
</page>
```

HANDS ON! - Esempio1 XSL a

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="page">
      <html>
         <head>
            <title>
               <xsl:value-of select="title"/>
            </title>
         </head>
         <body bgcolor="#ffffff">
            <xsl:apply-templates/>
         </body>
      </html>
  </xsl:template>
```

HANDS ON! - Esempio1 XSL b

```
<xsl:template match="paragraph">
  <p align="center">
    <i>
      <xsl:apply-templates/>
    </i>
  </p>
</xsl:template>
</xsl:stylesheet>
```

HANDS ON! - Esempio1 Xalan

Let us use the Apache XSLT processor: Xalan.

1) Get Xalan from xml.apache.org/xalan/index.html

2)Set CLASSPATH=%CLASSPATH%;…/xalan.jar; …/xerces.jar

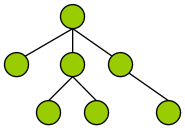
3) java **org.apache.xalan.xslt.Process**
–IN testPage.xml –XSL testPage.xsl –O out.html

HANDS ON! - Esempio1 Output HTML

```
<html>
  <head>
    <title>
      Test Page
    </title>
  </head>
  <body bgcolor="#ffffff">
    <p align="center">
      <i>
        What you see is what you get!
      </i>
    </p>
  </body>
</html>
```

The process

- The process starts by traversing the document tree, attempting to find a single matching rule for each visited node.
- Once the rule is found, the body of the rule is istantiated
- Further processing is specified with the *<xsl:apply-templates>*. The nodes to process are specified in the *match* attribute. If the attribute is omitted, it continues with the next element that has a matching template.



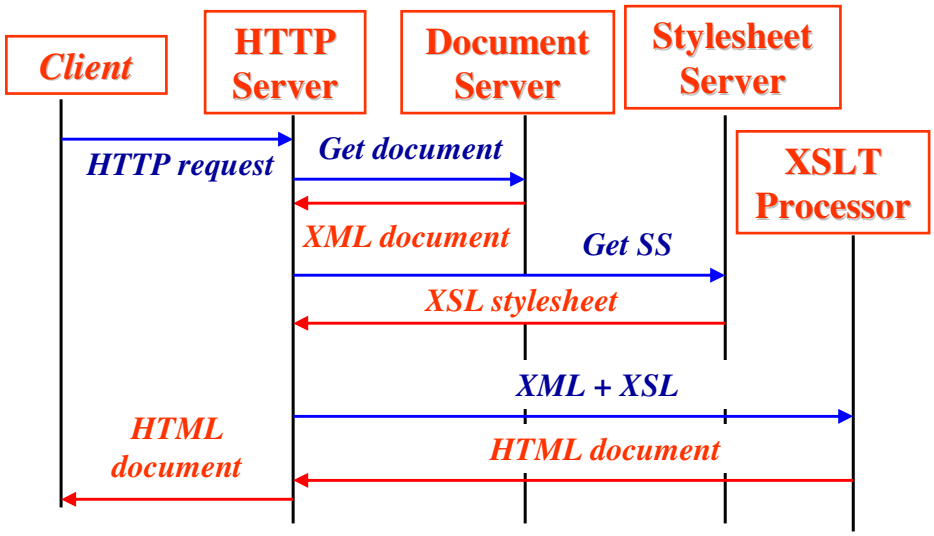
Implicit rules

```
<template match="/|*">
  <apply-templates/>
</template>

<template match="text()">
  <value-of select="."/>
</template>
```

Publishing frameworks

XML Enabled HTTP Server



Publishing frameworks

Apache Cocoon	http://xml.apache.org
Enhydra Application Server	http://www.enhydra.org/
Bluestone XML Server	http://www.bluestone.com/xml
SAXON	http://users.iclway.co.uk/mhkay/saxon