Web Services

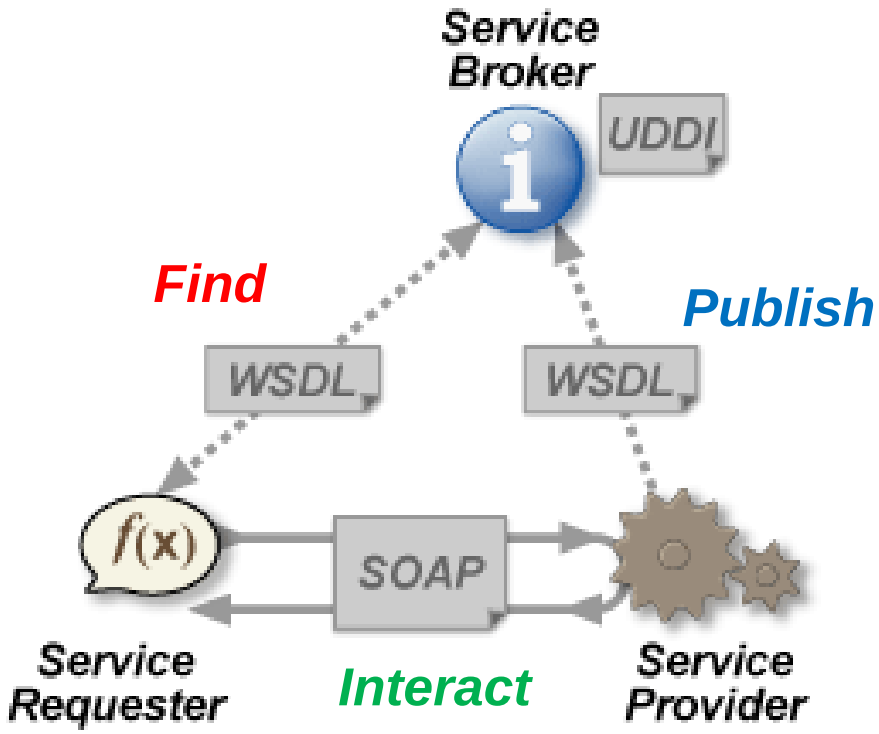
(XML applications)

Web Engineering

utad UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

Web Services

application communication over the Internet



identified by a Uniform Resource Identifier (URI), described and defined using XML

used to deliver interactive web services

SOAP : Simple Object Access Protocol

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

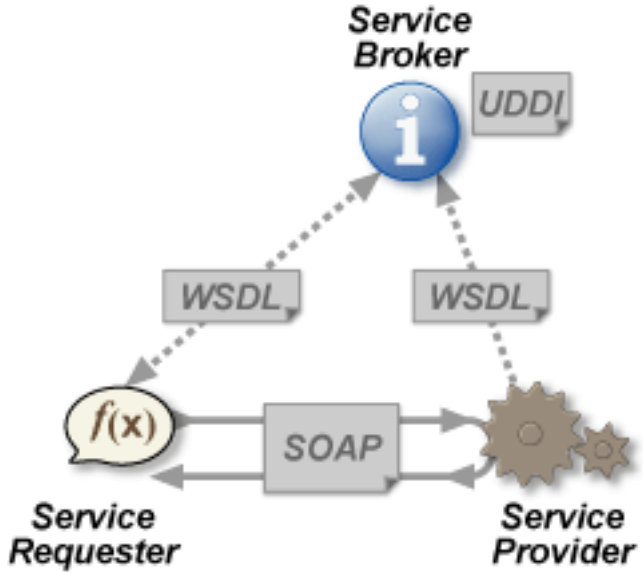
Request example

SOAP : Simple Object Access Protocol

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

Response example

SOAP : Simple Object Access Protocol



- Web Services Definition Language (WSDL)
  Is a XML document that describes a Web service. It specifies the location of the service and the operations (methods) the service exposes

- Universal Description, Discovery and Integration (UDDI)
  is a directory service where businesses can register and search for Web services.

WSDL Document:

In addition to describing each service, it also describes how they can be found. Its major elements are:

```
<types>: Defines the (XML Schema) data types used by the web service.

<message>: Defines the data elements for each operation.

<portType>: Describes the operations that can be performed and the messages involved.

<binding>: Defines the protocol and data format for each port type.
```

WSDL Document (example):

The following code is where the services are defined:

```
<wsdl:portType name="ICalculator">
  <wsdl:operation name="Add">
    <wsdl:input wsaw:Action="http://Example.org/ICalculator/Add"
                                    message="tns:ICalculator_Add_InputMessage" />
    <wsdl:output wsaw:Action="http://Example.org/ICalculator/AddResponse"
                                    message="tns:ICalculator_Add_OutputMessage" />
  </wsdl:operation>

  <wsdl:operation name="Subtract">
    <wsdl:input wsaw:Action="http://Example.org/ICalculator/Subtract"
                                    message="tns:ICalculator_Subtract_InputMessage" />
    <wsdl:output wsaw:Action="http://Example.org/ICalculator/SubtractResponse"
                                    message="tns:ICalculator_Subtract_OutputMessage" />
  </wsdl:operation>
</wsdl:portType>
```

WSDL Document (example):

The following code describes how each service should be called:

```
<wsdl:binding name="DefaultBinding_ICalculator" type="tns:ICalculator">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />7

   <wsdl:operation name="Add">
      <soap:operation soapAction="http://Example.org/ICalculator/Add" style="document" />
      <wsdl:input>
         <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
         <soap:body use="literal" />
      </wsdl:output>
   </wsdl:operation>

   <wsdl:operation name="Subtract">
      <soap:operation soapAction="http://Example.org/ICalculator/Subtract" style="document" />
      <wsdl:input>
         <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
         <soap:body use="literal" />
      </wsdl:output>
   </wsdl:operation>

</wsdl:binding>
```

WSDL Document (example):

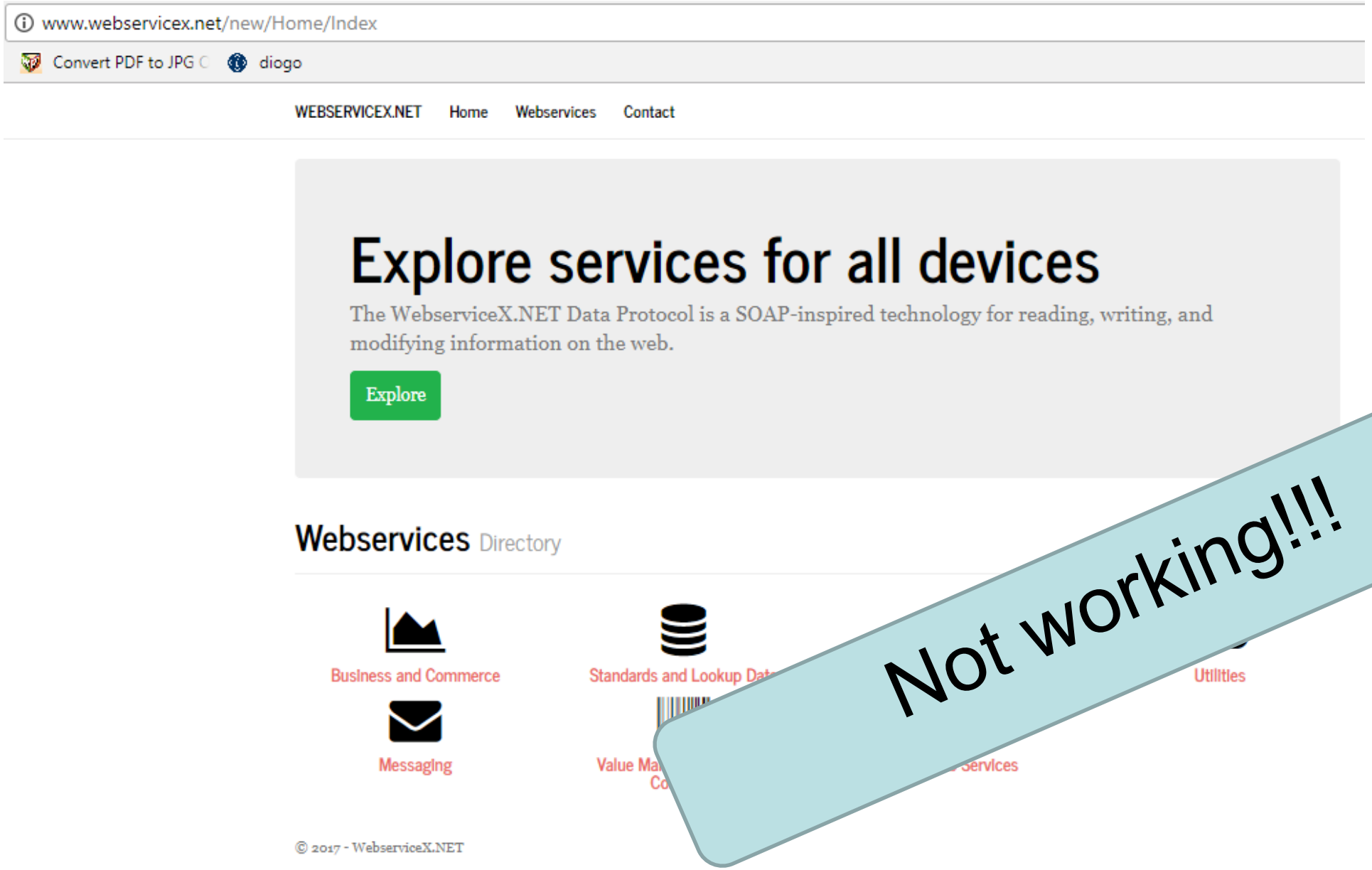
The following code defines the location of the *CalculatorService* service:

```
<wsdl:service name="CalculatorService">
   <wsdl:port name="ICalculator" binding="tns:DefaultBinding_ICalculator">
      <soap:address location="http://Example.org/ICalculator" />
   </wsdl:port>
</wsdl:service>
```

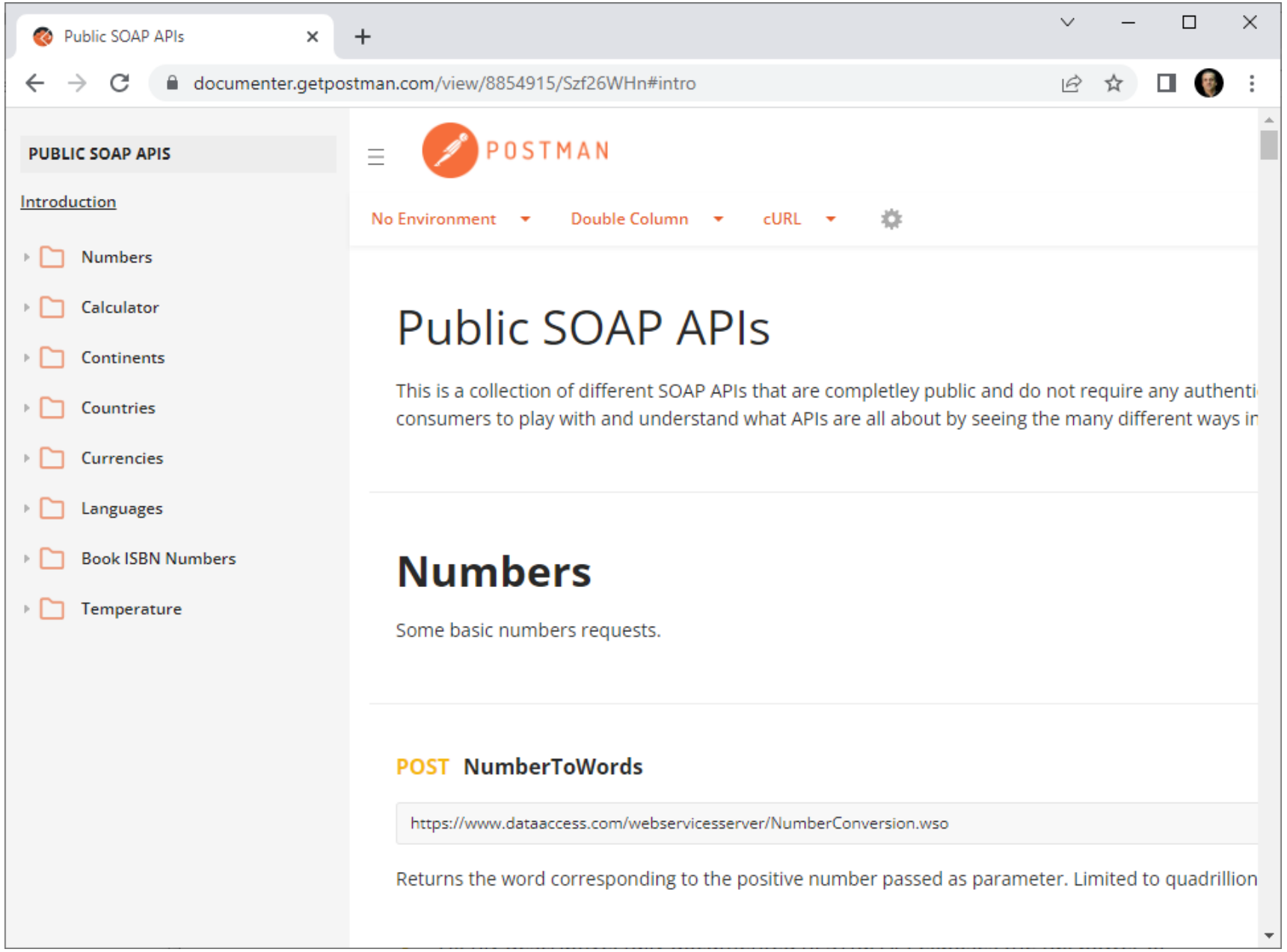
Another WSDL Document (example):

https://www.tutorialspoint.com/wsdl/wsdl_example.htm

Directory of public SOAP Web Services



Example of public SOAP Web Service



Webservices : RESTful Systems

REST : Representational State Transfer

SOAP vs REST
(example)

SOAP

```
POST / HTTP/1.1
Host: www.acme.com

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
                  soap-encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:body pb="http://www.acme.com/phonebook">
    <pb:GetUserDetails>
           <pb:UserId>12345</pb:UserId>
    </pb:GetUserDetails>
  </soap:body>
</soap:Envelope>
```

REST

```
GET /phonebook/UserDetails/12345 HTTP/1.1
Host: www.acme.com
```

Webservices : RESTful Systems

REST : Representational State Transfer

HTTP Methods and Their Meaning

Method	Meaning
GET	Read data
POST	Insert data
PUT or PATCH	Update data, or insert if a new id
DELETE	Delete data

REST vs SOAP

REST

RESTs sweet spot is when you are exposing a public API over the internet to handle CRUD operations on data. REST is focused on accessing named resources through a single consistent interface.

SOAP

SOAP brings it's own protocol and focuses on exposing pieces of application logic (not data) as services. SOAP exposes operations. SOAP is focused on accessing named operations, each implement some business logic through different interfaces.

REST vs SOAP

REST

Areas that REST works really well for are:

- **Limited bandwidth and resources**; remember the return structure is really in any format (developer defined). Plus, any browser can be used because the REST approach uses the standard GET, PUT, POST, and DELETE verbs. Again, remember that REST can also use the XMLHttpRequest object that almost modern browsers support today, which adds an extra bonus of AJAX.
- **Totally stateless operations**; if an operation needs to be continued, then REST is not the best approach and SOAP may fit it better. However, if you need stateless CRUD (Create, Read, Update, and Delete) operations, then REST is it.
- **Caching situations**; if the information can be cached because of the totally stateless operation of the REST approach, this is perfect.

REST vs SOAP

SOAP

Areas that SOAP is a great solution:

- **Asynchronous processing and invocation**; if your application needs a guaranteed level of reliability and security then SOAP 1.2 offers additional standards to ensure this type of operation. Things like WSRM – WS-Reliable Messaging.
- **Formal contracts**; if both sides (provider and consumer) have to agree on the exchange format then SOAP 1.2 gives the rigid specifications for this type of interaction.
- **Stateful operations**; if the application needs contextual information and conversational state management then SOAP 1.2 has the additional specification in the WS structure to support those things (Security, Transactions, Coordination, etc).

Styles and transformations: XSL and XSLT

(XML applications)

Web Engineering

utad UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

Consider the following XML document…

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="simple.xsl"?>
<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>
       two of our famous Belgian Waffles
    </description>
    <calories>650</calories>
  </food>
  <food>
    <name>Strawberry Belgian Waffles</name>
    <price>$7.95</price>
    <description>
       Light Belgian waffles covered with strawberries and whipped cream
    </description>
    <calories>900</calories>
  </food>
</breakfast_menu>
```

…and the next XSL

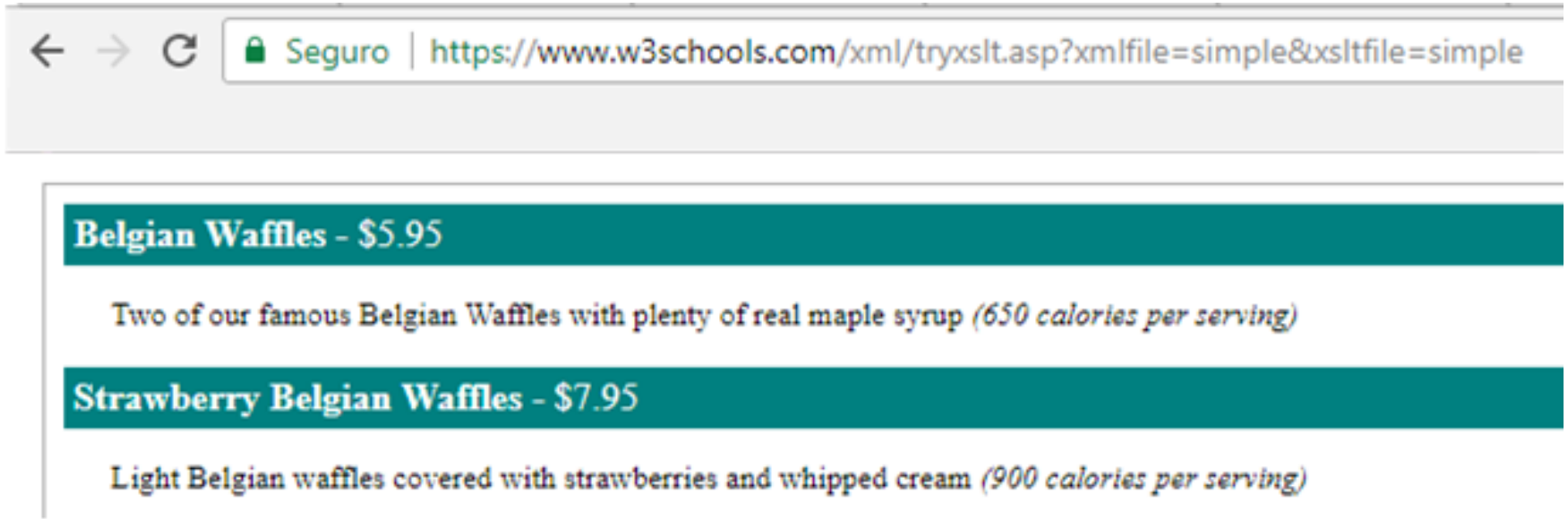
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<html xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="http://www.w3.org/1999/xhtml">
  <body style="font-family:Arial;font-size:12pt;background-color:#EEEEEE">
    <xsl:for-each select="breakfast_menu/food">
      <div style="background-color:teal;color:white;padding:4px">
        <span style="font-weight:bold"><xsl:value-of select="name"/></span>
        - <xsl:value-of select="price"/>
      </div>
      <div style="margin-left:20px;margin-bottom:1em;font-size:10pt">
        <xsl:value-of select="description"/>
        <span style="font-style:italic">
          <xsl:value-of select="calories"/> (calories per serving)
        </span>
      </div>
    </xsl:for-each>
  </body>
</html>
```

XSLT : XSL Transformations

XSL : *eXtensible Stylesheet Language*

What is the result of the XSLT (transformation)?

https://www.w3schools.com/xml/tryxslt.asp?xmlfile=simple&xsltfile=simple

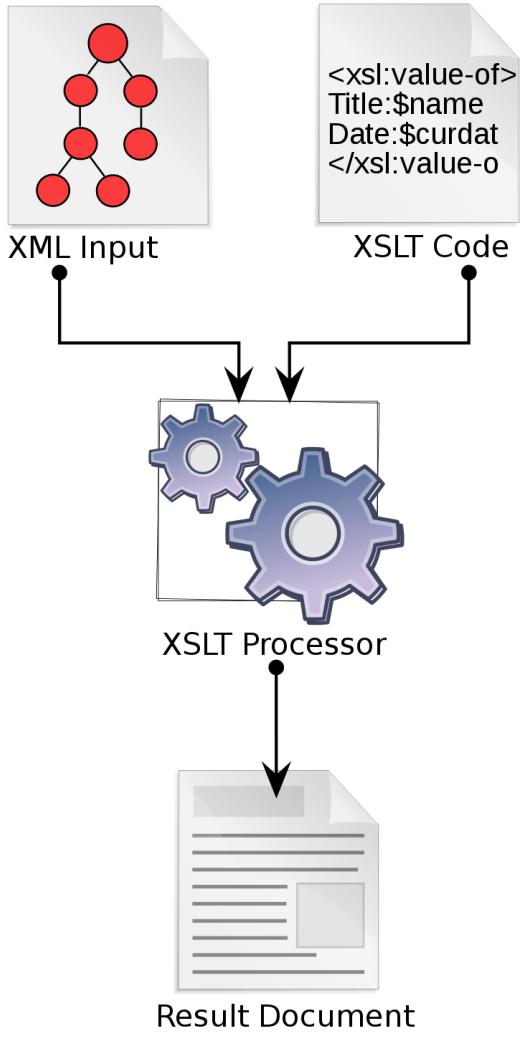


eXtensible Stylesheet Language (XSL)

serves the dual purpose of transforming XML documents and of exhibiting control over document rendering

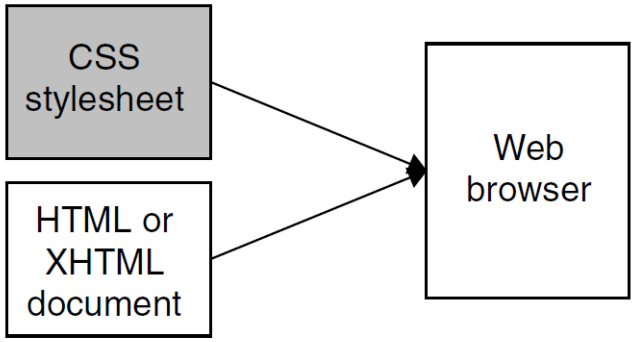
Transformation component of XSL (XSLT)

makes it possible to select fragments of XML documents, based on path patterns in the element hierarchy, and to apply transformation operations to these fragments

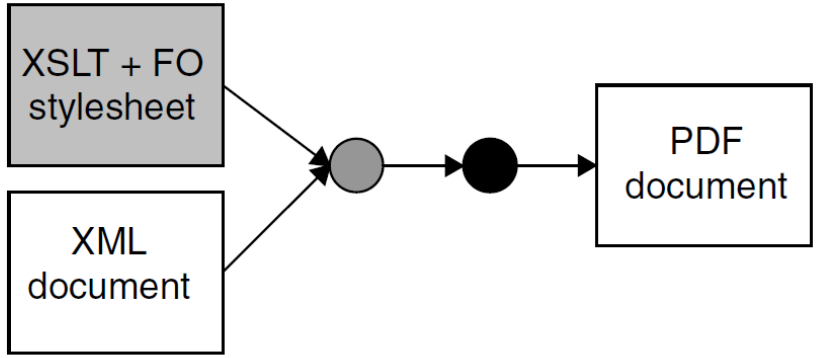


XSL Formating Objects (XSLT-FO)

markup language that describes the rendering
vocabular designed to support pagination



Using CSS stylesheets to render HTML and XHTML documents



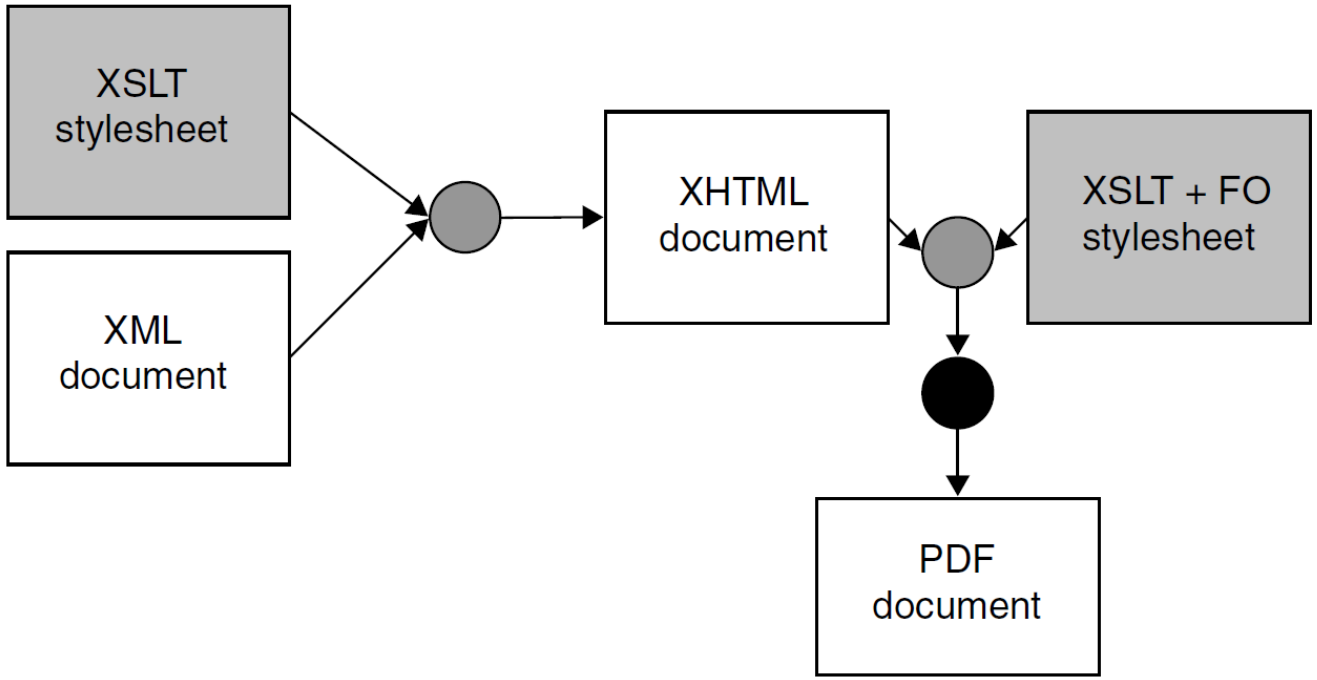
Using XSL stylesheets to print XML documents

XSL Formating Objects (XSLT-FO)

```
P { font: italic bold 12pt/14pt Times, serif; color: #0000F0  }
```

```
<fo:block font-size="12pt" font-weight="bold">content</fo:block>
```

XSL Formating Objects (XSLT-FO)



Alternate pattern for using XSL stylesheets to print XML documents

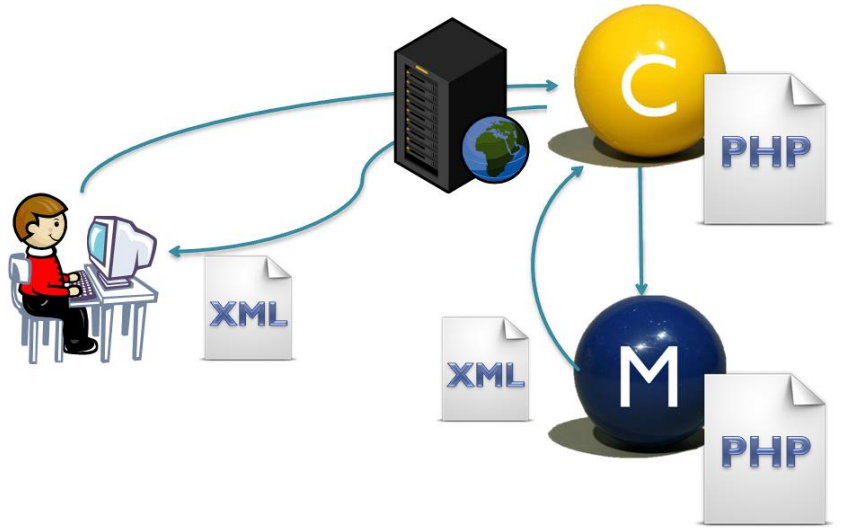
XPath - XML Path Language

query language for selecting nodes from an XML document

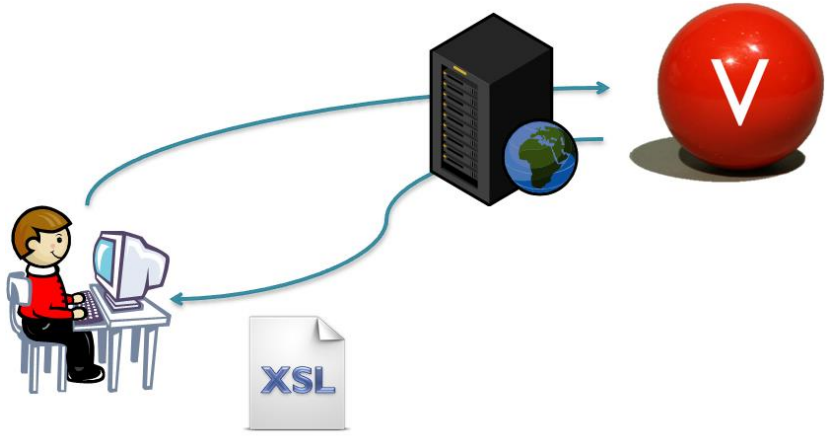
XSLT uses XPath to identify subsets of the source document tree and perform calculations

How important is it to use XSL to define the visual aspect of data?

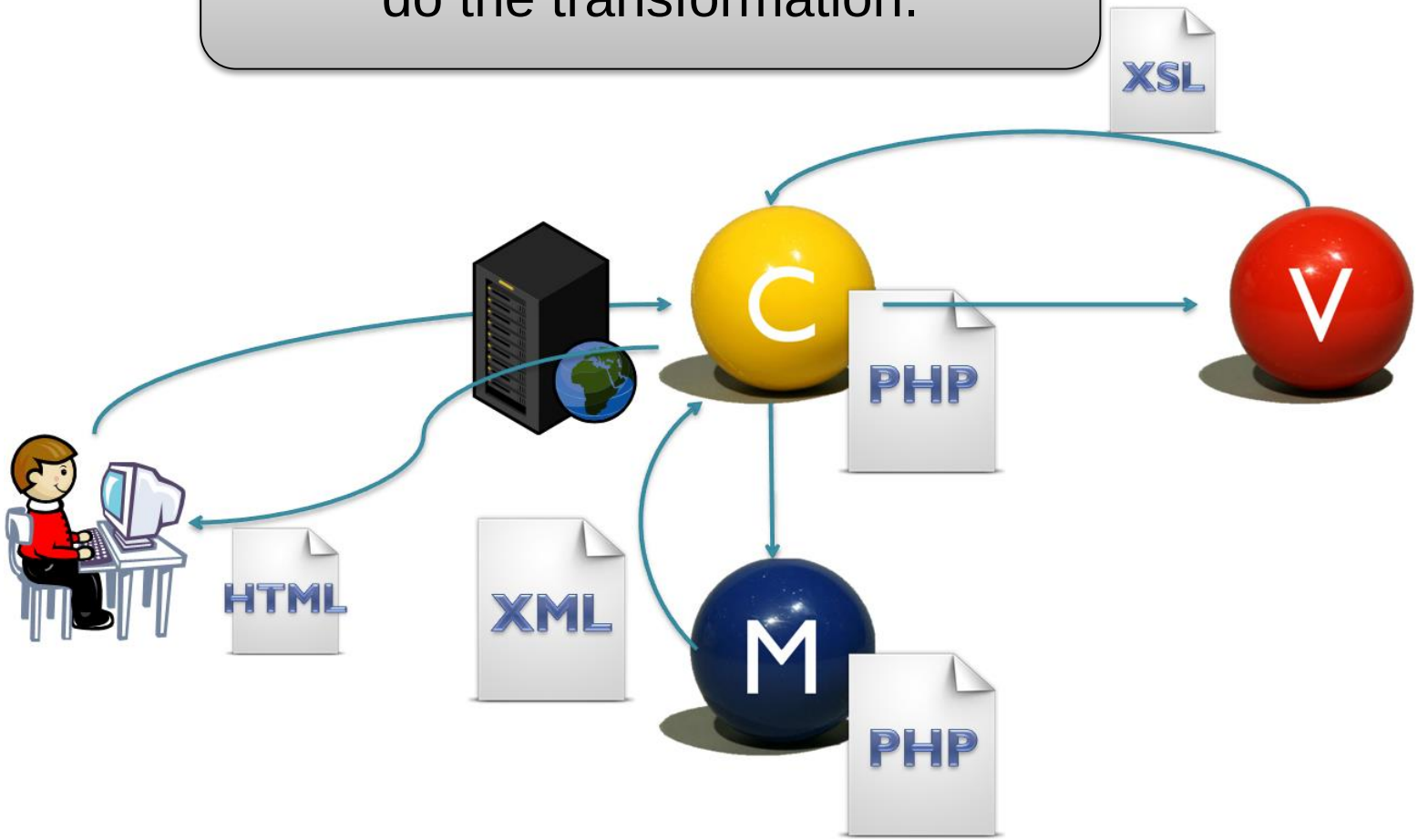
Step 1: User interacts with browser, sending request to Controller



Step 2: Browser automatically prompts for the View XSL

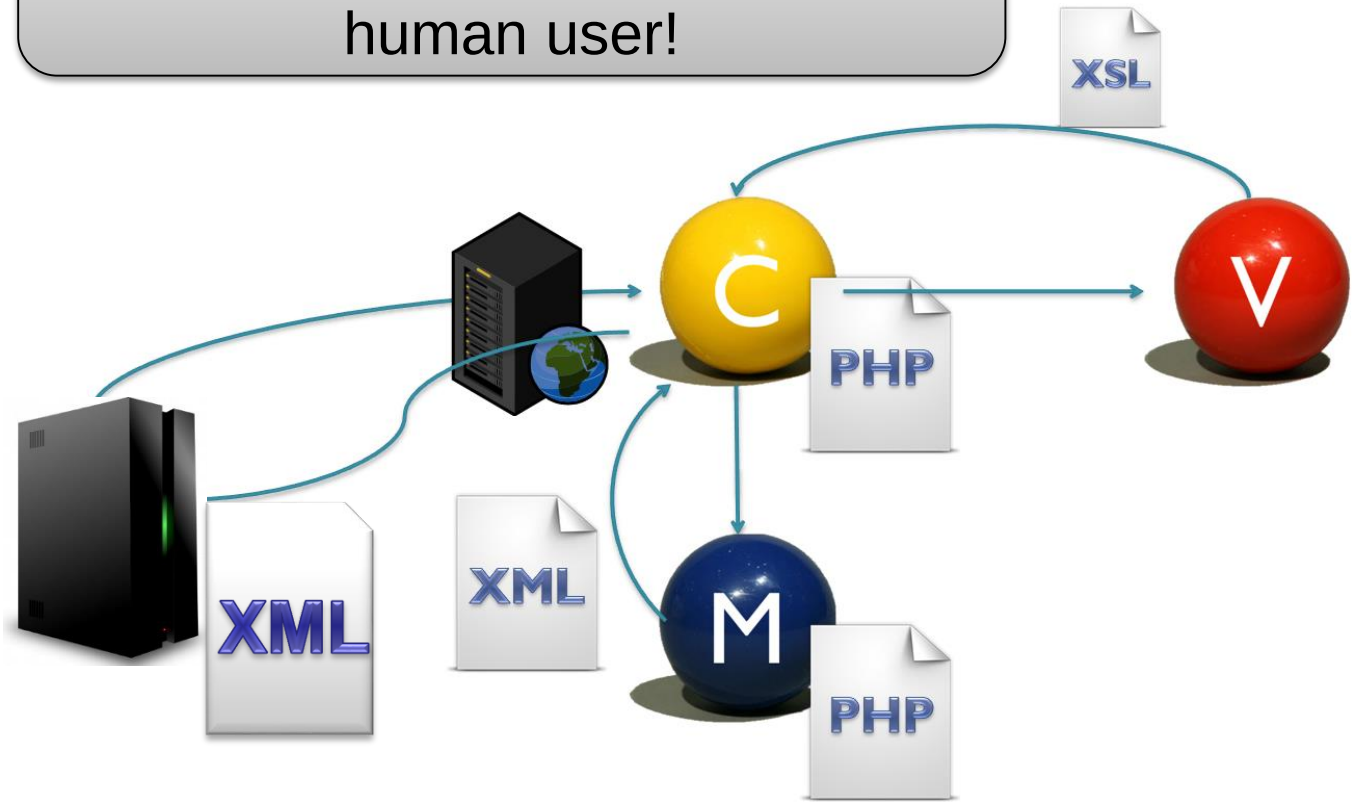


... or Controller uses View XSL to do the transformation.



Why is it important that we can use XSL to convert files without visuals?

Let us remember that the destination does not have to be a human user!



Alternatives to XSL? …CSS

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="css.css"?>
<schedule>
  <date>Tuesday 20 June</date>
  <programme>
    <starts>6:00</starts>
    <title>News</title>
    With Michael Smith and Fiona Tolstoy.
    Followed by Weather with Malcolm Stott.
  </programme>
  <programme>
    <starts>6:30</starts>
    <title>Regional news update</title>
    Local news for your area.
  </programme>
  <programme>
    <starts>7:00</starts>
    <title>Unlikely suspect</title>
    Whimsical romantic crime drama starring Janet
    Hawthorne and Percy Trumpp.
  </programme>
</schedule>
```

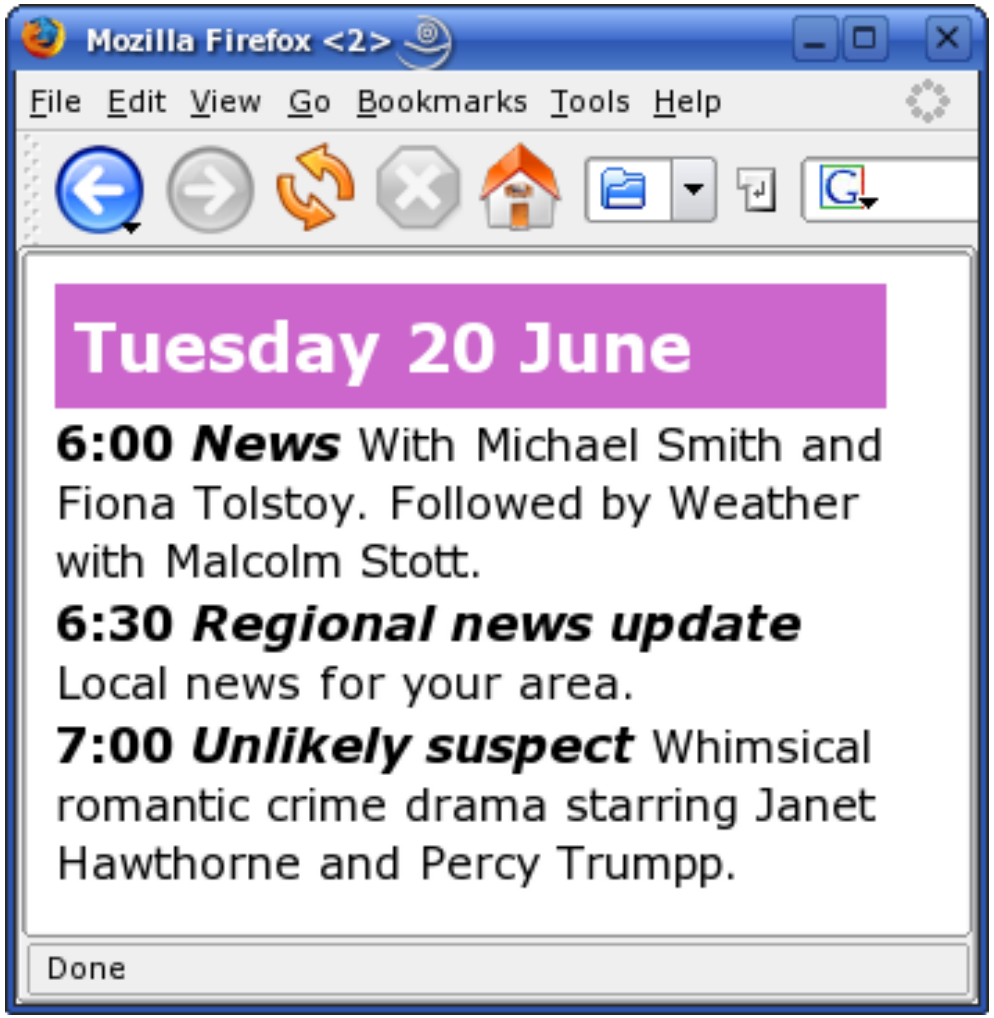
+

```css
@media screen {
  schedule {
    display: block;
    margin: 10px;
    width: 300px;
  }
  date {
    display: block;
    padding: 0.3em;
    font: bold x-large sans-serif;
    color: white;
    background-color: #C6C;
  }
  programme {
    display: block;
    font: normal medium sans-serif;
  }
  programme > * { /* All children of programme elements */
    font-weight: bold;
    font-size: large;
  }
  title {
    font-style: italic;
  }
}
```

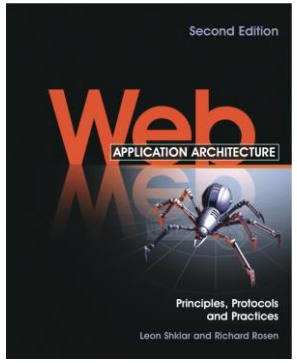
Alternatives to XSL? …CSS

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="css.css"?>
<schedule>
  <date>Tuesday 20 June</date>
  <programme>
    <starts>6:00</starts>
    <title>News</title>
    With Michael Smith and Fiona Tolstoy.
    Followed by Weather with Malcolm Stott.
  </programme>
  <programme>
    <starts>6:30</starts>
    <title>Regional news update</title>
    Local news for your area.
  </programme>
  <programme>
    <starts>7:00</starts>
    <title>Unlikely suspect</title>
    Whimsical romantic crime drama starring Janet
    Hawthorne and Percy Trumpp.
  </programme>
</schedule>
```

```css
ns-serif;

;

ans-serif;
}
programme > * { /* All children of programme elements */
  font-weight: bold;
  font-size: large;
}
title {
  font-style: italic;
}
}
```



Bibliography



Shklar, Leon & Rosen, Rich (2009). Web Application Architecture: Principles, Protocols and Pratices. Chichester, Reino Unido: John Wiley & Sons.

Pages: 100 to 119

Chapter 5

5.3 Web Services

5.4 XSL