DTD: Example

```
<!DOCTYPE TVSCHEDULE [
    <!ELEMENT TVSCHEDULE (CHANNEL+)>
    <!ELEMENT CHANNEL (BANNER,DAY+)>
    <!ELEMENT BANNER (#PCDATA)>
    <!ELEMENT DAY (DATE,(HOLIDAY|PROGRAMSLOT+)+)>
    <!ELEMENT HOLIDAY (#PCDATA)>
    <!ELEMENT DATE (#PCDATA)>
    <!ELEMENT PROGRAMSLOT (TIME,TITLE,DESCRIPTION?)>
    <!ELEMENT TIME (#PCDATA)>
    <!ELEMENT TITLE (#PCDATA)>
    <!ELEMENT DESCRIPTION (#PCDATA)>

    <!ATTLIST TVSCHEDULE NAME CDATA #REQUIRED>
    <!ATTLIST CHANNEL CHAN CDATA #REQUIRED>
    <!ATTLIST PROGRAMSLOT VTR CDATA #IMPLIED>
    <!ATTLIST TITLE RATING CDATA #IMPLIED>
    <!ATTLIST TITLE LANGUAGE CDATA #IMPLIED>
    ]>
```

XML Schema

- XML Schema is an XML-based alternative to DTD.
- An XML schema describes the structure of an XML document.
- The XML Schema language is also referred to as XML Schema Definition (XSD).
- purpose of an XML Schema is to define the legal building blocks of an XML document, just like a DTD.

An Example...

```
<?xml version="1.0"?>
<xs:schema>
    <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>
<note>
<to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

The <schema> Element

```
<?xml version="1.0"?>

  <xs:schema>
  ...
  ...
  </xs:schema>
```

What is a Simple Element?

A simple element is an XML element that can contain only text. It cannot contain any other elements or attributes.

The syntax for defining a simple element is:

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

where xxx is the name of the element and yyy is the data type of the element.

XML Schema has a lot of built-in data types. The most common types are:

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

Example

- Here are some XML elements:

```
<lastname>Fernandes</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

- And here are the corresponding simple element definitions:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

Default and Fixed Values for Simple Elements

- A default value is automatically assigned to the element when no other value is specified.

In the following example the default value is "red":

```
<xs:element name="color" type="xs:string" default="red"/>
```

- A fixed value is also automatically assigned to the element, and you cannot specify another value.

In the following example the fixed value is "red":

```
<xs:element name="color" type="xs:string" fixed="red"/>
```

How to Define an Attribute?

- The syntax for defining an attribute is:

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

where xxx is the name of the attribute and yyy specifies the data type of the attribute.

XML Schema has a lot of built-in data types. The most common types are:

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

- **Example**
- Here is an XML element with an attribute:

<lastname lang="EN">Smith</lastname>

- And here is the corresponding attribute definition:

<xs:attribute name="lang" type="xs:string"/>

What is a Complex Element?

A complex element is an XML element that contains other elements and/or attributes.

There are four kinds of complex elements:

- empty elements
- elements that contain only other elements
- elements that contain only text
- elements that contain both other elements and text

How to Define a Complex Element

- Look at this complex XML element, "employee", which contains only other elements:

```
<employee>
    <firstname>John</firstname>
    <lastname>Smith</lastname>
</employee>
```

- We can define a complex element in an XML Schema as:
- The "employee" element can be declared directly by naming the element, like this:

```
<xs:element name="employee">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="firstname" type="xs:string"/>
        <xs:element name="lastname" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
```

XML Schema: An Example

```
<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>
```

External Schema

```
<?xml version="1.0"?>

<note xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="http://www.w3schools.com
note.xsd">

<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Internal Schema

```
<?xml version="1.0"?>
<xs:schema>
    <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>
<note>
<to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

Displaying XML with CSS

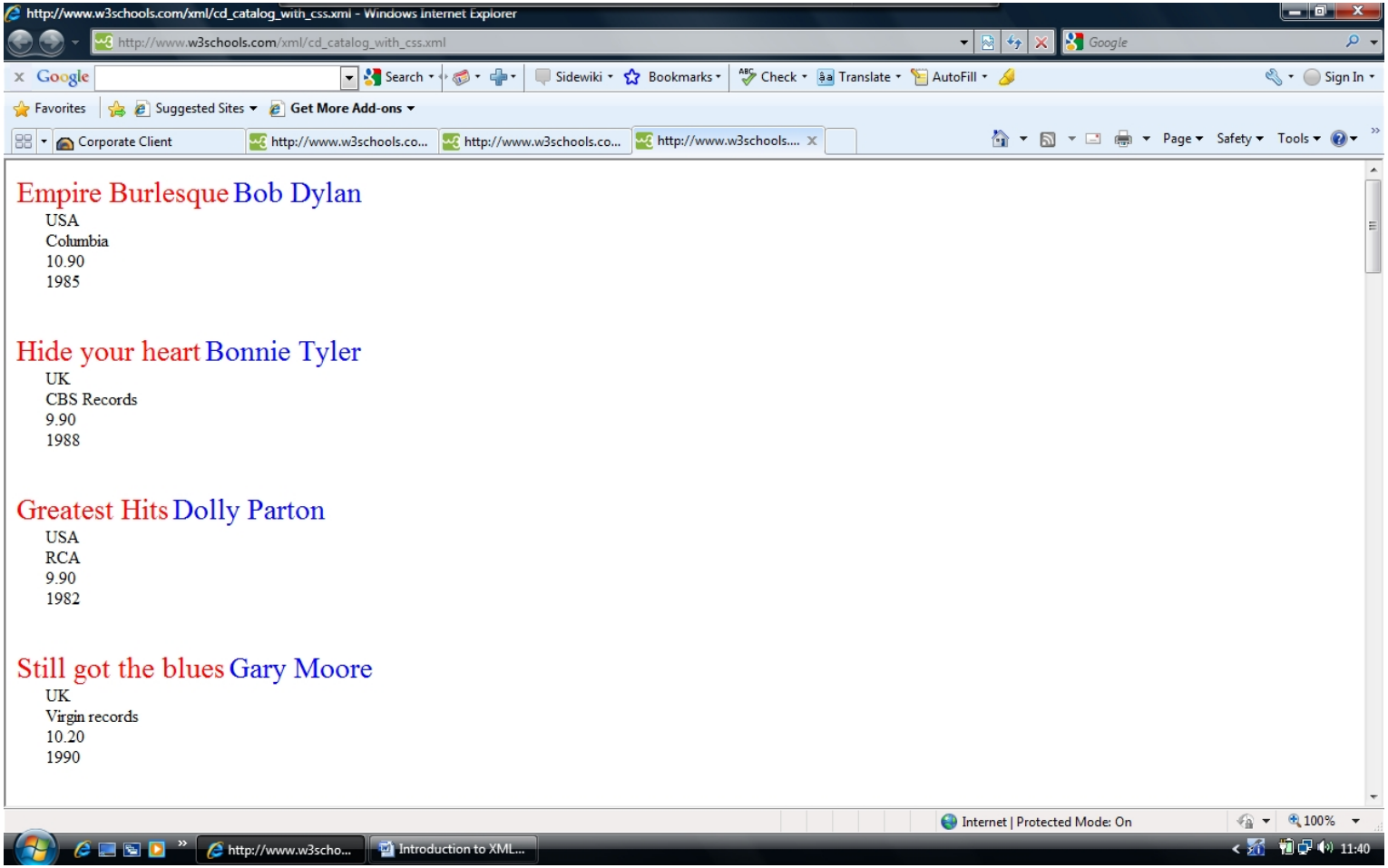
Take a look at this XML file: The CD catalog

```
<?xml version="1.0" ?>
- <CATALOG>
    - <CD>
        <TITLE>Empire Burlesque</TITLE>
        <ARTIST>Bob Dylan</ARTIST>
        <COUNTRY>USA</COUNTRY>
        <COMPANY>Columbia</COMPANY>
        <PRICE>10.90</PRICE>
        <YEAR>1985</YEAR>
    </CD>
    - <CD>
        <TITLE>Hide </TITLE>
        <ARTIST>Bonnie Tyler</ARTIST>
        <COUNTRY>UK</COUNTRY>
        <COMPANY>CBS Records</COMPANY>
        <PRICE>9.90</PRICE>
        <YEAR>1988</YEAR>
      </CD>
</CATALOG>
```

Then look at this style sheet: The CSS file

```
CATALOG
{
background-color: #ffffff;
width: 100%;
}
CD
{
margin-bottom: 30pt;
margin-left: 0;
}
TITLE
{
color: #FF0000;
font-size: 20pt;
}
ARTIST
{
color: #0000FF;
font-size: 20pt;
}
COUNTRY,PRICE,YEAR,COMPANY
{
color: #000000;
margin-left: 20pt;
}
```

Finally, view: The CD catalog formatted with the CSS file



Linking CSS with XML

```
<?xml version="1.0" ?>
    <?xml-stylesheet type="text/css" href="cd_catalog.css"?>
    <CATALOG>
     <CD>
      <TITLE>Empire Burlesque</TITLE>
      <ARTIST>Bob Dylan</ARTIST>
      <COUNTRY>USA</COUNTRY>
      <COMPANY>Columbia</COMPANY>
      <PRICE>10.90</PRICE>
      <YEAR>1985</YEAR>
     </CD>
     <CD>
      <TITLE>Hide your heart</TITLE>
      <ARTIST>Bonnie Tyler</ARTIST>
      <COUNTRY>UK</COUNTRY>
      <COMPANY>CBS Records</COMPANY>
      <PRICE>9.90</PRICE>
      <YEAR>1988</YEAR>
     </CD>
    .
    .
    .
    </CATALOG>
```

Displaying XML with XSLT

- XSLT is the recommended style sheet language of XML.
- XSLT (eXtensible Stylesheet Language Transformations) is far more sophisticated than CSS.
- XSLT can be used to transform XML into HTML, before it is displayed by a browser
- **It Started with XSL**
- **XSLT Uses XPath**
- XSLT uses XPath to find information in an XML document. XPath is used to navigate through elements and attributes in XML documents.
- All major browsers have support for XML and XSLT.

We want to **transform** the following XML document ("cdcatalog.xml") into XHTML:

```
<?xml version="1.0" ?>
  <catalog>
    <cd>
      <title>Empire Burlesque</title>
      <artist>Bob Dylan</artist>
      <country>USA</country>
      <company>Columbia</company>
      <price>10.90</price>
      <year>1985</year>
    </cd>
  .
  .
  </catalog>
```

Then create an XSL Style Sheet ("cdcatalog.xsl") with a transformation template:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
 <html>
 <body>
  <h2>My CD Collection</h2>
  <table border="1">
   <tr bgcolor="#9acd32">
    <th>Title</th>
    <th>Author</th>
   </tr>
   <xsl:for-each select="catalog/book">
   <xsl:sort select="title"/>
   <tr>
    <td><xsl:value-of select="title"/></td>
    <td><xsl:value-of select="author"/></td>
   </tr>
   </xsl:for-each>
  </table>
 </body>
 </html>
</xsl:template>

</xsl:stylesheet>
```

Link the XSL Style Sheet to the XML Document

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="cd.xsl"?>
<catalog>
  <book id="bk101">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <price>44.95</price>
    <publish_date>2000-10-01</publish_date>
    <description>An in-depth look at creating applications
    with XML.</description>
  </book>
</catalog>
```

The <xsl:template> Element

- The <xsl:template> element is used to build templates.
- The **match** attribute is used to associate a template with an XML element. The match attribute can also be used to define a template for the entire XML document. The value of the match attribute is an XPath expression (i.e. match="/" defines the whole document).

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

<xsl:template match="/">
 <html>
 <body>
 <h2>My CD Collection</h2>
 <table border="1">
  <tr bgcolor="#9acd32">
   <th>Title</th>
   <th>Author</th>
  </tr>
  <tr>
   <td>.</td>
   <td>.</td>
  </tr>
 </table>
 </body>
 </html>
</xsl:template>

</xsl:stylesheet>
```

- The **<xsl:template>** element defines a template. The **match="/"** attribute associates the template with the root of the XML source document.
- The content inside the <xsl:template> element defines some HTML to write to the output.
- The last two lines define the end of the template and the end of the style sheet.

The result from this example was a little disappointing, because no data was copied from the XML document to the output.

XSLT <xsl:value-of> Element

The <xsl:value-of> element is used to extract the value of a selected node

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
 <html>
 <body>
 <h2>My CD Collection</h2>
 <table border="1">
  <tr bgcolor="#9acd32">
   <th>Title</th>
   <th>Author</th>
  </tr>
  <tr>
   <td><xsl:value-of select="catalog/cd/title"/></td>
   <td><xsl:value-of select="catalog/cd/author"/></td>
  </tr>
 </table>
 </body>
 </html>
</xsl:template>

</xsl:stylesheet>
```

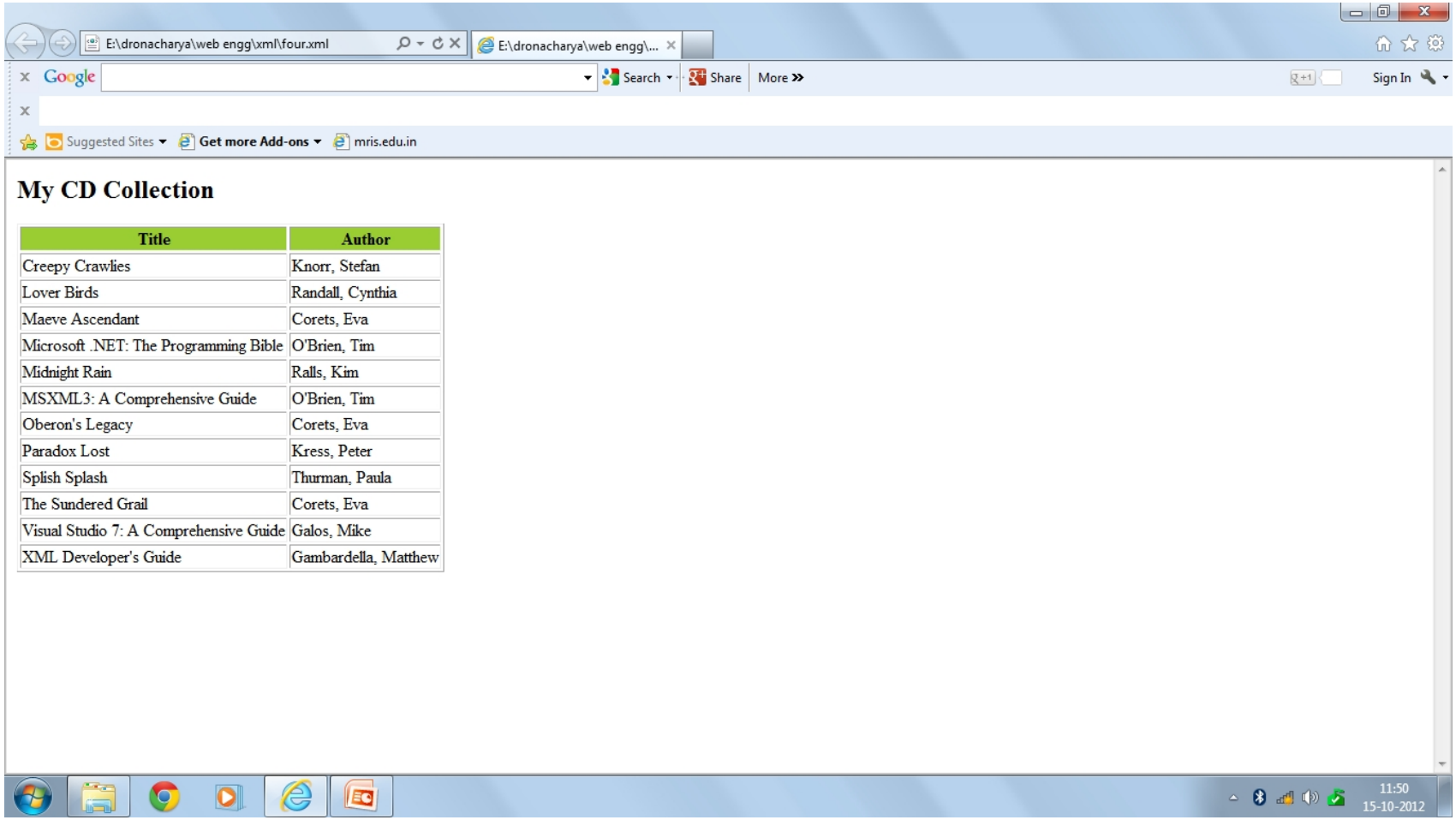
XSLT <xsl:for-each> Element

The <xsl:for-each> element allows you to do looping in XSLT

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
 <html>
 <body>
 <h2>My CD Collection</h2>
 <table border="1">
  <tr bgcolor="#9acd32">
   <th>Title</th>
   <th>Artist</th>
  </tr>
  <xsl:for-each select="catalog/cd">
  <tr>
   <td><xsl:value-of select="title"/></td>
   <td><xsl:value-of select="author"/></td>
  </tr>
  </xsl:for-each>
 </table>
 </body>
 </html>
</xsl:template>

</xsl:stylesheet>
```


E:\dronacharya\web engg\xml\four.xml
E:\dronacharya\web engg\...
Google
Search
Share
More
Sign In
Suggested Sites
Get more Add-ons
mris.edu.in
My CD Collection
Title
Author
Creepy Crawlies
Knorr, Stefan
Lover Birds
Randall, Cynthia
Maeve Ascendant
Corets, Eva
Microsoft .NET: The Programming Bible
O'Brien, Tim
Midnight Rain
Ralls, Kim
MSXML3: A Comprehensive Guide
O'Brien, Tim
Oberon's Legacy
Corets, Eva
Paradox Lost
Kress, Peter
Splish Splash
Thurman, Paula
The Sundered Grail
Corets, Eva
Visual Studio 7: A Comprehensive Guide
Galos, Mike
XML Developer's Guide
Gambardella, Matthew
11:50
15-10-2012