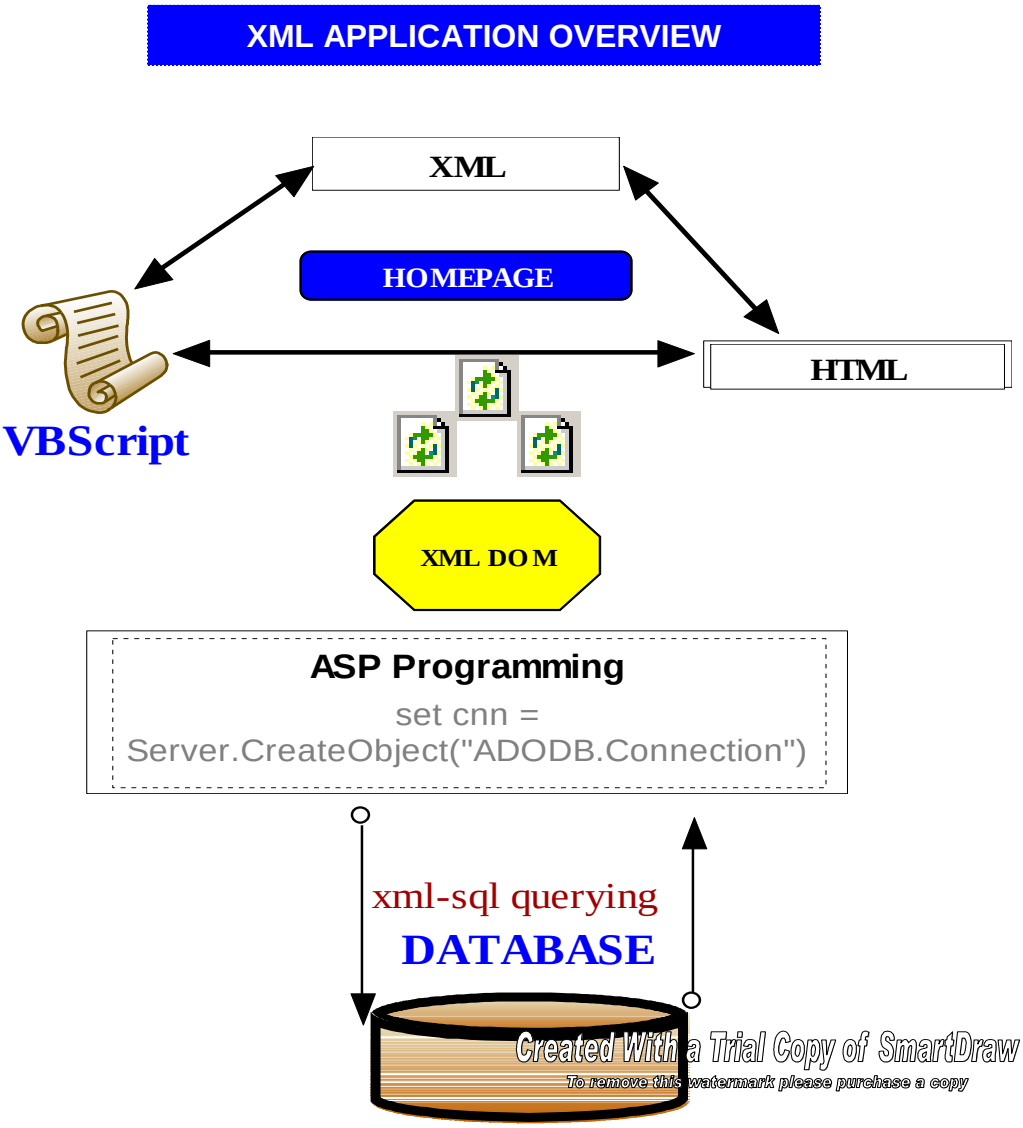
eXtensible Markup Language

An extensive study

Submitted to
Prof.Ausif Mahmood
University of Bridgeport

Student Name: Mahesh Babu Gandra
Student number: 456012



SCHEMATIC REPRESENTATION OF THE APPLICATION

(Smart Draw software is used to draw the pictorial representation above)

INTRODUCTION:

This is an online data-tracking tool, which takes inputs from the user. The Front End is designed (HTML) and includes XML Data islands. This application also has modified, delete and clear data directly from the homepage (Front-End). I have used an Access database as a back-end database.

Since this application has been developed by me very recently, until the number of user and data storage increases (very soon), I would be requesting for tables in ORACLE. So I have included the option to switch between SQL and access databases.

Detailed description with each every command is dealt in the coding part of the thesis. (Please refer power points & coding section)

Homepage Development:

I have developed an HTML page with the tables and frames (please refer homepage). And called JavaScript to highlight the selected records. Whenever the mouse rolls-over the data, the corresponding record will be highlighted. The main functionally of this application is DATAISLANDS. User would be able to see the Total Backend on the homepage through the Front-end. So that User has a comprehensive exposure to all records stored in the backend.

This has been achieved using XML & Visual Basic interface. Commands like xmldso.recordset.AbsolutePosition to interact with XML data islands created by Microsoft DOM call-ups.

The functionalities like UPDATE, INSERT, DELETE & CLEAR would be discussed extensively below.

Dealing with RECORDS

Before creating any file/function, connection files should be configured, Where the connection to the database is established with set cnn = Server.CreateObject ("ADODB.Connection"). And xml function file which creates the data islands for the existing records (access data base) is created with the function Set objNode = objDom.createElement (strName), which determined the various nodes starting from child node with appending functionality. Each and every command is clearly described in the coding section of the application. And also each value typed in from the front is validated by arCpt(GC_Dos) = oNode.dos.nodeTypedValue. Similarly all XML function has been created (Available in XML func.asp file).

Schema has been decided to establish the connection between the front-end and backend. Two layers of xml functionality, one between front and backend, another one is an interface between connection asp file and access database.

Note, in the all following files, xmlfunction.fle and conn.file should be used as include files.

Every file is described separately with basic simple steps.

INSERT.asp

Connection has been configured as we discussed above.

1. Data has been extracted from the values from the incoming XML (front –end interface) with command strCustomerID= GetNodeValue (objDOM, "/Supplier/Supplier ID").
2. Building the Update SQL statement strSQL = strSQL & "Insert Into Supplier". similar queries have been used to call the remaing values.
3. Finally the queries are executed with cnnNW.Execute strSQL
4. After executing the query, the main would be directed to confirmation Page.
5.

MODIFY. asp

1. Set objDOM = Server.CreateObject ("Microsoft.XMLDOM") is used to Create an XML DOM from the incoming XML. DOM function is described in the earlier part of the thesis.
2. 'Extract the values from the incoming XML object, strCustomerID = GetNodeValue (objDOM, "/Supplier/Supplier ID")

3. Building the SQL update Query with the command strSQL = strSQL & " Update Supplier Set ", similar for remaining records.

4. Executing the updated Sql query cnnNW.Execute strSQL.

DELETE.asp

1.Like the other folders step1, Creating an XML DOM from the incoming XML set objDOM = Server.CreateObject("Microsoft.XMLDOM") objDOM. Load(Request)

2. Defining the Delete SQL statement: strCustomerID= GetNodeValue (objDOM, "/Supplier/Supplier ID") for eliminating the respecting fields selected in the front –end.

3.Executing the Sql query established above to remove the data from database.

Discussing about XML process every time the above function is executed.

Each function is breakdown in simple steps as below:

1. Set the Connection to the database using Dim cnnXML, strXML, objXML, objXSL.

2. Obtain the XML data from the specified (supplier ID) database using strXML = GetXMLData (cnnXML, '').

3. Creating an instance of the XML documents set objXML = Server.CreateObject ("Microsoft.XMLDOM").

5. Merging XML files with XLT files.
   Using the command Set objXSL = Server.CreateObject ("Microsoft.XMLDOM")

6. Loading the XML object with XML from the database using ObjXML.async = false
   ObjXML.loadXML strXML

7. Loading the XSL from the XSL file using objXSL.async = false and
   ObjXSL.load Server.MapPath ("NWDetails.xsl")

Sending Data to the front end on request basis

1.Initiating the XML transferring functionality at the beginning with objXML.transformNodeToObject objXSL, Response.

2.Determine the system functions from the previously called XMLfunction file with Function GetXMLData (cnnXML, strDataType) and dim rsXML, strXML, strSQL, stStream; strSupplier ID, DQ, Since the Primary key of the database table is Supplier ID. Such as strSupplier ID = Request ("cID").

3. Using the Loop to send every time the data is requested by the user.

Do While Not rsXML.EOF
Loop
strXML = strXML & "</root>"

4.Supplying the Front end tables with data such as GetXMLData = strip (strXML)

For elaborate explanation of each and every single function please refer the program files section.

Conclusion:

This application proves XML as the excellent backend technology for storing and sharing data in a highly structured manner. And also demonstrates XML provides the framework for creating customized markup languages, as we have customized for our data needs in the application discussed above. So the data would have the flexibility of spreadsheet along with the visual accessibility of a web page. The XML files are platform and browser independent, So Irrespective of the platform standard the above function would work with same setups and behavior of data modeling.

Extensive study on XML is presented in the study part of this thesis.

An Extensive study on XML

PREFACE

Before I start up the detailed description of the thesis, I would like to thank Professor. Ausif Mahmood for his kind time extended to me in his busy schedules.He has provided me a valuable feedback , whenever I have requested him. I sincerely appreciate his time and guidance provided to me throughout my studies.

The main focus of this thesis is on the XML applicability in **e**commerce. I have divided thesis into two sections. Section one focuses on fundamentals to advanced XML programming concepts, coding and designing. Section two is description of an application developed by me which is an online tool, tracks the imperative information provided by the client.

Application, online Supplier management tool is a supplier information tracking tool, which was designed using ASP, VBScirpt and XML technologies. And Since the tool is newly developed by me, I have the backend database maintained in the Access database. As the data grows , I would be migrating this data to ORACLE.

As for the XML description in the first section, I have manifested a different kind of examples for different events/components. And also included the screenshots wherever required. Attached powerpoints provides more comprehensive description

SECTION A

TABLE OF CONTENTS:

SECTION B is described in POWERPOINT presentation.

Descriptive view of XML

Introduction

Xml is a general mark-up language that is used to structure data. More specifically, XML is a general mark up language for any number of reasons. XML code looks a lot like HTML code, except that the tag and attribute names are completely customized to fit each different type of data being represented. More important , is the fact the XML documents are very easy to process. This implies that automated applications such as web search engines can easily sift through XML documents and extract considerably more meaning than they can get from HTML documents.

XML ,unlike, HTML is an extremely broad data-structuring standard that has implications far beyond web pages. For example, consider this question, HTML is to webpages as XML is to what? This is a difficult question, since XML isn't really geared toward any one solution. Instead XML provides the framework for creating customized solutions to a wide range of problems. This made possible by XML base markup languages, which are custom markup languages that we created using XML.

XML is a meta-language, which is a fancy way of saying that it is a language used to create other mark-up language. Using XML, unique markup language can be created to model just about any kind of information, including page content.

HTML takes care of the format, needless to say. The key point is that XML lays the ground rules for organizing information in a consistent manner.

To understand the need for XML, you have to first consider the role of HTML. In the primitive days of the internet, some programmers created HTML by simplifying another mark-up language such as SGML. What is SGML? . A detailed description is given in the attached powerpoints with comparisation and convergence to XML).

However, there are a few additional observations to make. Firstly, though technology is changing rapidly and is exerting great influence on the way we do business, the latter tends to level off some of the former's extravagances.

1 The XML Language : A First Outline

You may have heard about the World Wide Web Consortium (W3C) for instance, the standards body that has developed this new technology. As prerequisite to XML, one should definitely get an awareness rotation to Atleast one markup language, is that the origin of XML is in the world of SGML, the Standard Generalized Markup Language. SGML is an ISO standard that defines an extremely powerful markup language, and for many years has been used in the publishing industry and large manufacturing companies.. Being a meta language, SGML is used to create other markup languages such as HTML.

A comparison between XML and HTML

To better understand XML and its relationship to HTML, we need to know where HTML created problems. HTML was originally designed as a means of sharing written ideas among scientific researchers. So in its inception, HTML was never intended to support fancy graphics, formats, ir page layout features. Instead, HTML was intended to focus on the meaning of information or the content of information. It wasn't until web browser vendors got excited that HTML was expanded to address the presentation of information.
Although presentation plays an important role in any website, modern web applications are evolving to become driven by data of very specific types such as finanacial transactions. HTML is not a effective markup language for representing such data. With its support for custom markup languages, XML opens door for carefully describing data and the relationships between pices of data.

By focusing on content, XML allows description of the information in web documents. If XML describes data better than HTML, then does it mean that

XML , is set to outdate HTML as the markup language of choice for the web? no, it is not, XML is not a replacement for HTML, or even a competitor of HTML.XML's impact of HTML has to do more with cleaning up HTML than it does with altering HTML.The best way to compare XML and HTML is to remember that XML establishes a set of strict rules that any markup language must follow.

HTML is a relatively unstructured markup language that could benefit from the rules of XML. The natural merger of two technologies is to make HTML adhere to the rules and structure of XML To accomplish the merger, a new version of HTML has been formulated that adheres to the stricter rules of XML. The new XML-Complaint version of HTML is known as XHTML. XML's relationship with HTML doesn't end with XHTML, however XHTML is a great idea that may someday make WebPages cleaner and consistent for browsers to display, we are a ways off from seeing web designers embrace XHTML.

.

With XML, it is finally recognized that the information published on a website is usually more than text – it is data. Using a presentation-only markup language as HTML, it is quite obvious that much of the structure and meaning of this data is lost. To
overcome this and other limitations, the W3C (World Wide Web Consortium) has developed XML. The goal was to create a language that was easy to learn and use, compatible with SGML, powerful enough to have a wide variety of applications and
legible to both humans and computers.
Adding up all these features, we can build the following definition :
XML (eXtensible Markup Language) is a meta-markup language defined by the W3C, that provides a way to create extensible formats for describing structured data in electronic documents and to express rules about those data. It is a subset of SGML, optimized for delivery over the web, and its current version is 1.0 Second Edition.
An important detail in this definition is "meta", which means that it is a "language to
create other languages". Unlike HTML, which has a fixed tagset, XML has no tagset of
its own, but is used to define new tagsets and data structures, called XML applications1.
In a certain sense, there's no such thing as an 'XML document' – all the documents that
use XML-compliant syntax are really using applications of XML. Indeed, this also means that HTML can be described as an XML-application, which has been done in the

XHTML specification, a reformulation of HTML 4.0 in XML 1.0.

It should be noted that (X)HTML is only one type of application. So far, only matters
related to web publishing have been mentioned, where the fact that XML separates content from style is essential. However, when the XML-designers had a “wide variety
of applications” in mind, they were not only thinking of humans interacting with a web
server through a browser.

The other features of the XML language will be dealt later chapters below. One more aspect to discuss is electronic documents. While browsing the web can still be seen as reading documents, in the context of communication between computers it is more appropriate to talk about data structures. Since the ability of something like “common sense” is still rare to computers, a document like this, even nicely divided into paragraphs, has little meaning.

To expand our basic knowledge whatever we have gained from the above discussion, let’s go construct a example and discuss. Here' the realestate price listing with addresses:

```
<?xml version="1.0"?>
<REALESTATE.LISTINGS>
  <TITLE>Test of Real Estate data using XML and XSL</TITLE>
  <LISTING ZONED="Residential">
    <PRICE>99,000</PRICE>
    <STYLE>Victorian</STYLE>
    <STREET>113 Granger Road</STREET>
    <BEDROOMS>3</BEDROOMS>
  </LISTING>
  <LISTING ZONED="Residential">
    <PRICE>129,900</PRICE>
    <STYLE>Contemporary</STYLE>
    <STREET>635 Darlington Road</STREET>
    <BEDROOMS>4</BEDROOMS>
  </LISTING>
  <LISTING ZONED="Commercial">
    <PRICE>55,000</PRICE>
    <STYLE>Storefront</STYLE>
    <STREET>35 Main Street</STREET>
    <BEDROOMS>0</BEDROOMS>
  </LISTING>
```

</REALESTATE.LISTINGS>

The structure of the above code: starts with a definition of XML version , which is a syntax for every XML document. Next step is defines the parent tag. i.e starts with real estate listing and ends the parent tag with real estate. The other tags between the parent is called child nodes such as PRICE, STYLE, BED ROOM. The tags must be nested , yes it is a regular software practice to nest the loop. So XML goes that way.

And the in-between different tags are defined(user defined) ,Here we are defining our own tags, like real estate, street , listing and style. This is an great advantage of XML over HTML or any other markup language. These items are called elements. It's imperative to note that an element is a logical unit of information in an XML document, whereas a tag is a specific piece of information in an XML document, whereas a tag is specific piece of XML code that comprises an element. That's way the reference to an element is given by its name, such as TITLE, whereas tags are always referenced just as they appear in code such as <TITLE> or </TITLE>

Let's take a look at attribute? What is attribute?

Attributes are small pieces of information that appear within an elements opening tag. An attribute consists of an attribute name and a corresponding attribute value, which are separated by an equal symbol(=) . The value of an attribute appears to the right of the equal symbol and must appear within quotes. Following is san example of an attribute named name that is associated with the street element.

<STREET name="113 Granger Road/>
In this example, the name attribute is used to identify the name of a STREET. Attributes aren't limited to empty elements- they are just as useful with non empty elements.
< STREET name= "113 Granger Road" BEDROOMS="4" PRICE="55,000">
this is an example of how several attributes are used to describe the apartment in the particular STREET. Attributes are a great way to tie small pieces of information to an element.

As we have looked at the example , it's time to look and learn about the specific rules that govern its usage. The most important syntax are listed as below. There are a lot you will notice you to do advance program. I thought would be a good idea to capsule them in five.
Tags are case sensitive

2. Every opening tag must have a corresponding closing tags(unless it is abbreviated as an empty tag)
3. A nested tag pair cannot overlap another tag
4. Attribute values must appear within quotes

Every document must have a root element.

Our Example and applicability of rules:

Rule1 : <STYLE>Victorian</STYLE> and <style> </style> are different
Rule2: <PRICE> 125,000</PRICE> mentions that every opened tag should be closed.
Rule3: our example is clear nested between the STREET, PRICE and STYLE
Rule4: <STREET name=”113 Granger Road/> attributes are in quotes always.
Rule5:Every XML document must contain a root element, which means that only one element can be at the top level of any given XML document. In our example it is REALESTAT.LISTINGS.To a make a quick comparison with HTML, the HTML element in a web page is the root element, so HTML is similar to XML in this regard. However, technically HTML will let you get away with having more than one root element, whereas XML will not.

Unique XML symbols

There are a few special symbols in XML that must be entered differently than other text characters. The reason for entering these symbols differently is because they are used in identifying parts if an XML document such as tags and attributes. And also we should know a new term called entity here for further explanation. An entity is a symbol that identifies a resource , such as a text character or even a file. Let’s look at the predefined entities for the special characters.

Less than symbol(<) ---<
Greater than symbol(>)---&gt
Quote Symbol(“)--- "
Apostrophe Symbol (‘)--- '
Amphersand symbol (&)----&

Let’s look at the example
<apartment name=”10mill lane & PRICE's $125000”/>
This makes the attribute value a little tougher to read, but this is a good example of how XML is willing to make a exchange between ease of use on the developer’s part and technical clarity. Since there are only five predefined entities to deal with, so it’s easy to remember.

XML declaration

XML declartion is a line of code at the beginning of an xml document that identifies the version of XML used by the document. Currently there is only one version of XML (VERSION 1.0), there would be more features to be added up in future to get advanced features.
Following is the standard XML declaration for XML 1.0

<?xml version =”1.0”?>
This processing instruction is notifying the application that the document uses XML 1.0 . Processing instructions are easily identified by the <? And ?> symbols that are used to contain each instruction.

SELECTING AN xml editor.

In order to create and edit our own XML documents, we must have an application to server as an XML editor. Since XML documents are raw text documents, a simple text editor can serve as AN xml editor. For example,, a notepad. Im serious just a notepad.. If you want XML –specific features such as editing elements and attributes , we have to purchase a full blown XML editors.

CREATING OUR FIRST XML DOCUMENT

Creating an XML document is the first step in creating an XML based application. The directive we have to use to create an XML document is <?xml version =”version”?>

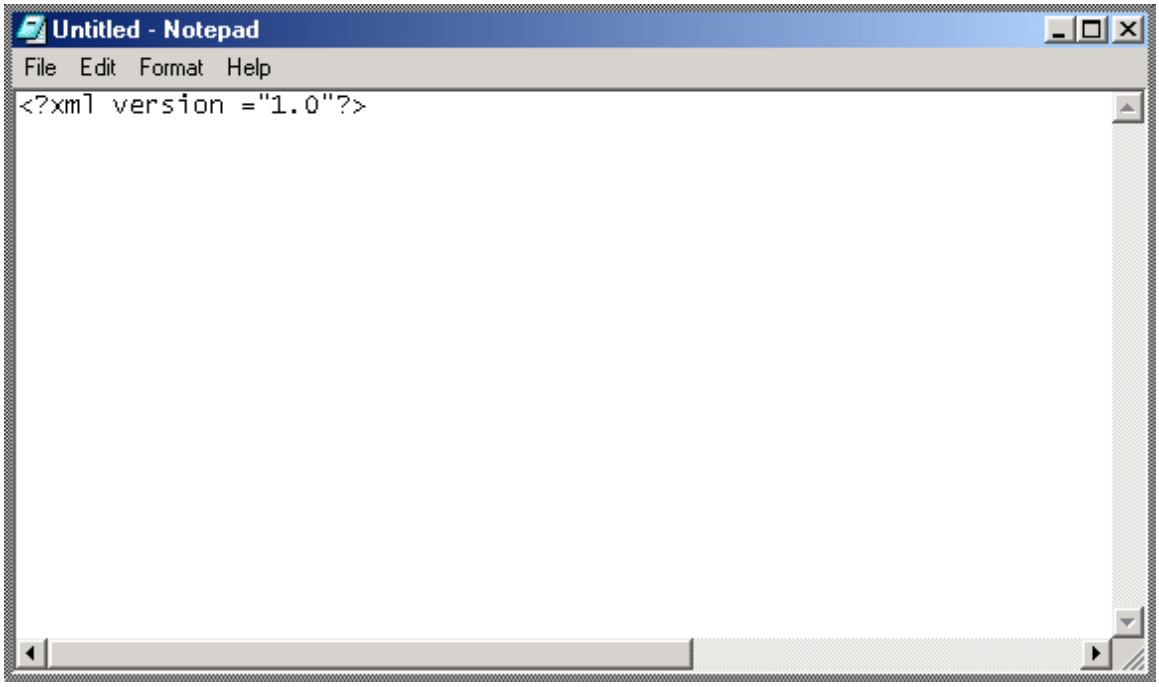


fig1.

[steps in creating XML document]

1.Create a new text file in the test editor of your choice. I took notepad.
2.Type the XML directive. Version as seen in fig.1
3. Type the beginning tag for an XML element as in fig.2 In this example the beginning XML tag is <REALESTATE.LISTINGS>

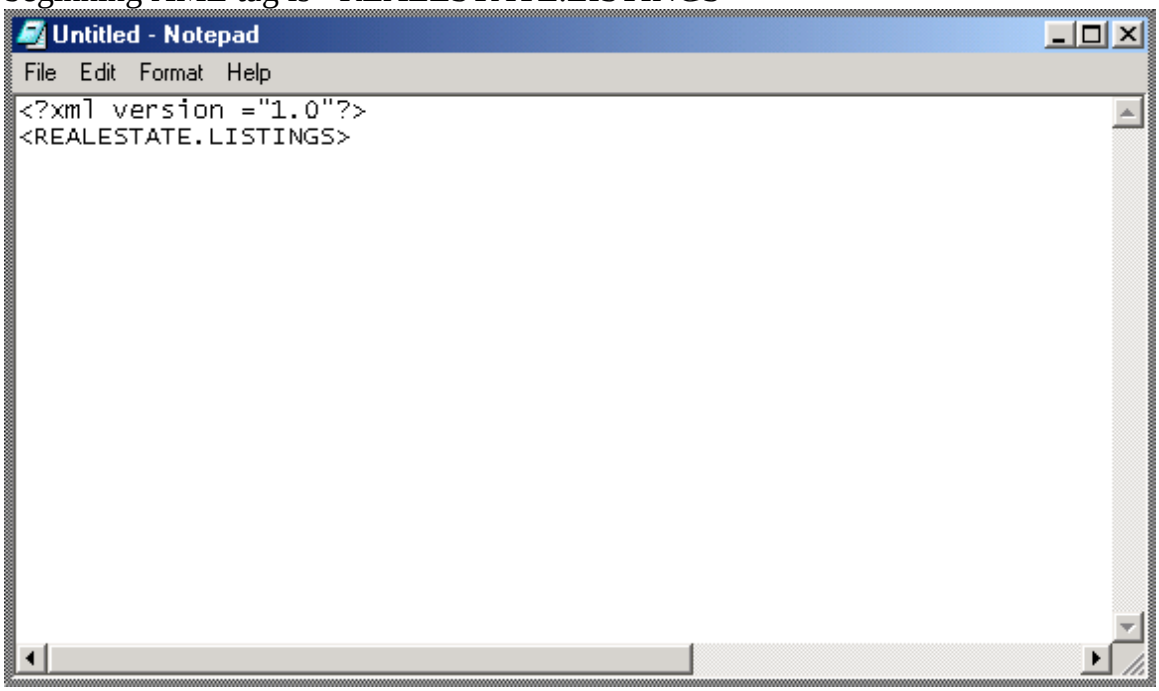


fig.2

4. Type the content for the XML element refer FIG.3 In our example the content is the matter between <REALESTATE.LISTING> and </REALESTATE.LISTINGS>

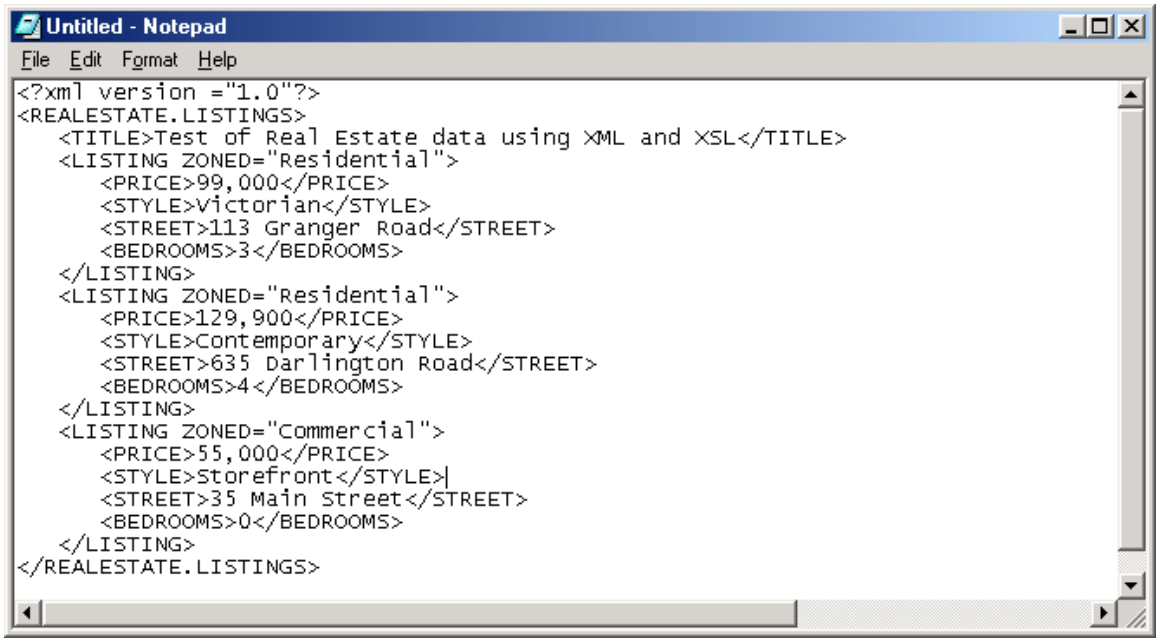


Fig3.

Let's save this file as realestate.xml in our local drive C.

Verification of our first XML document.

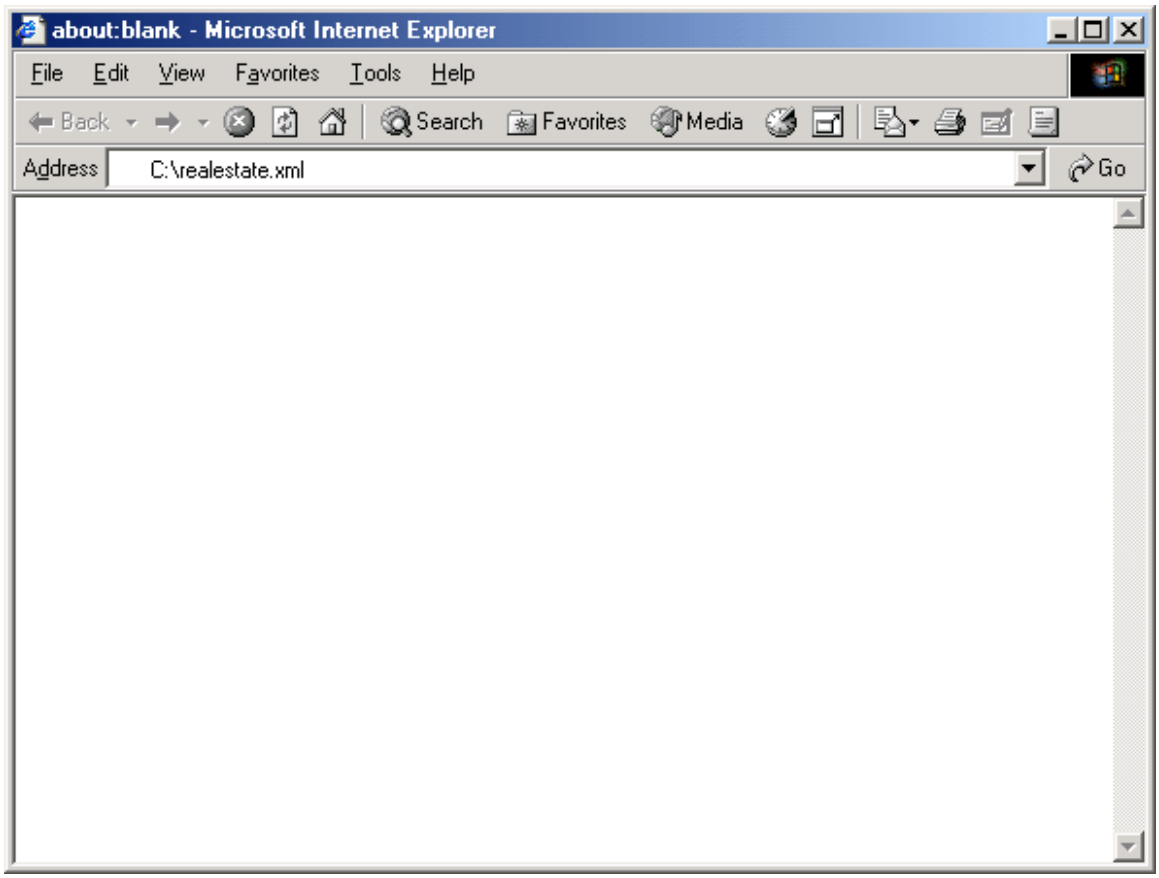


fig4.

Let's open a browser to locate our XML file. type in realestate.xml in the address bar.

And hit enter. We will see the following picture. Our XML document is displayed!!!.fig5.

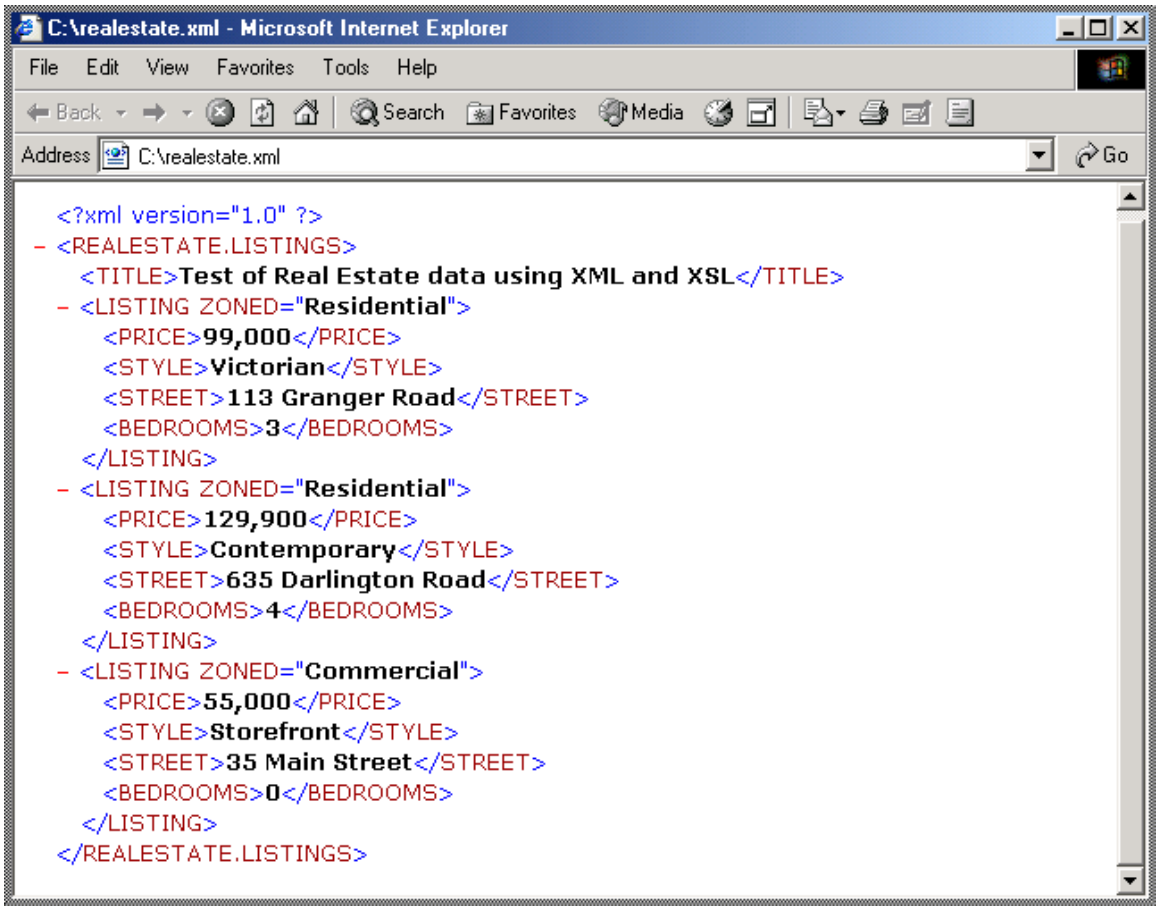


fig5.

One more step, we need a XML processor i.e code that accesses, manipulates, or diplay XML data intelligently. There are four general categories of XML processors.
Cascading Style Sheets(CSS). Simple XML data display. The easiest type of processors to create, and the most limited.
XSL style sheet. Sophisticated, dynamic XML data display
Data Island plus script. Incorporates XML data into an HTML presentation and performs some processing (such as error checking or data manipulation) in addition to display.
Data object model plus script or client-side program to create a full-blown XML application.

CREATE A XML processor

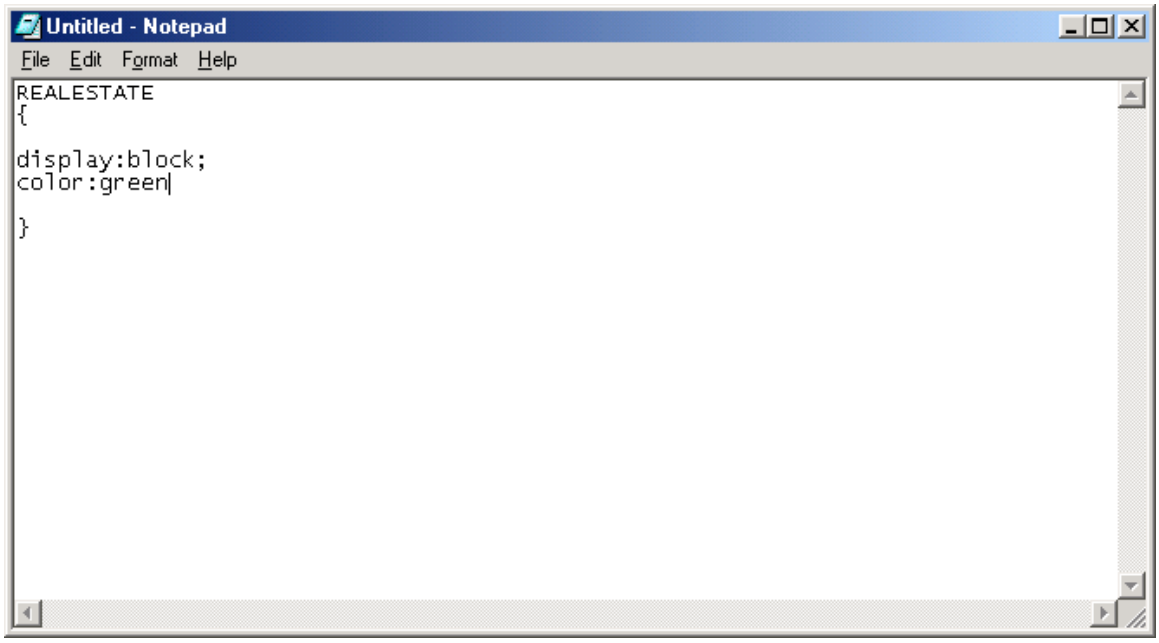


fig6.

Cascading style sheet is created, this file should be saved as .css extenstion. i.e realestate.css(store this file in C drive) In the example , the styles surrounded by curly braces are applied to an XML element named GREETING. Add a cascading style sheet directive to the XML document, specifying the cascading style sheet we have created.FIG7.

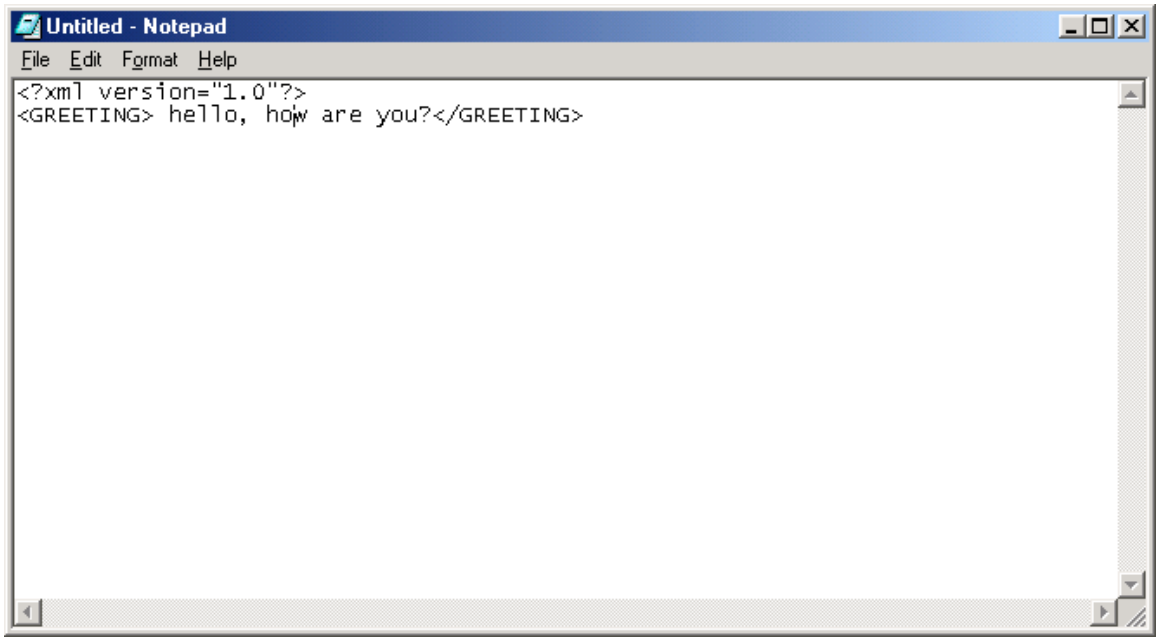


fig7.

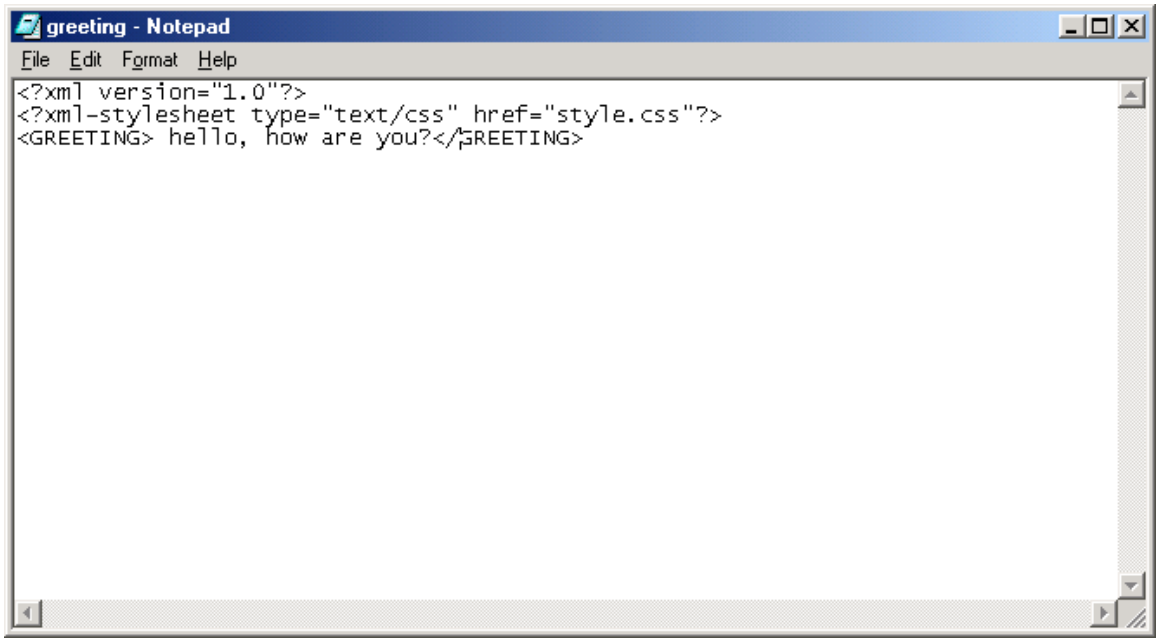


fig8.

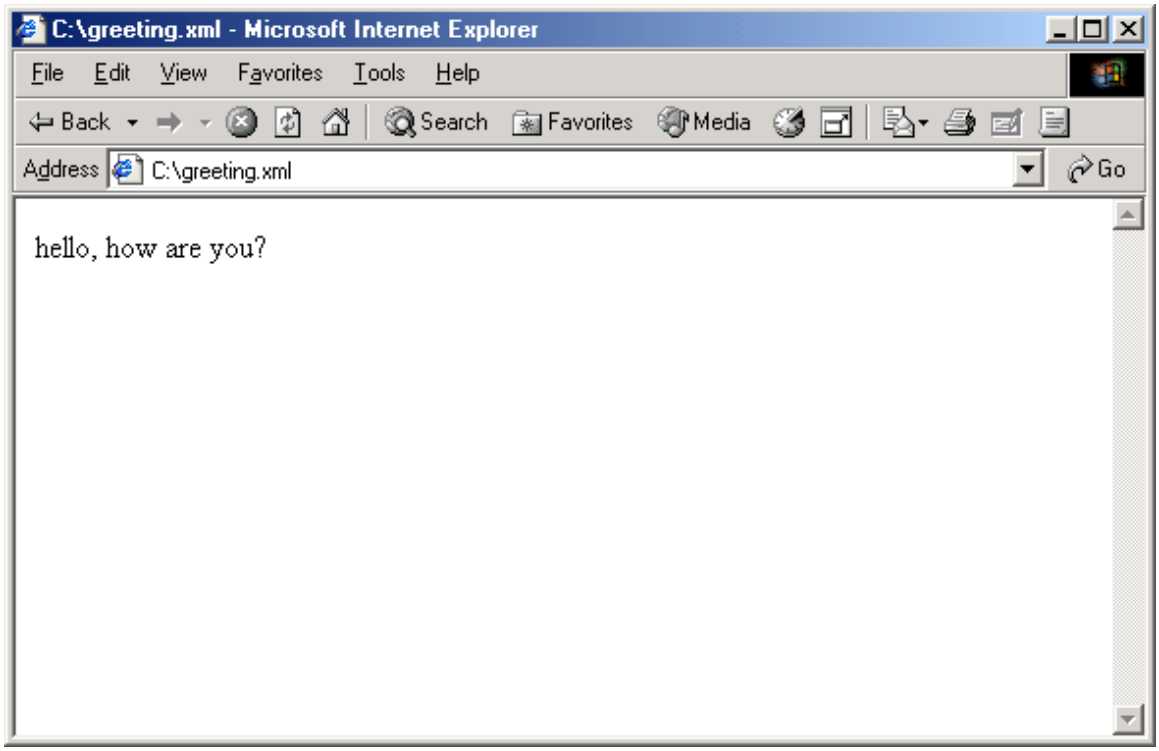


fig9.

The XML data displays based on the specified cascading style sheet rules. This pictures displays the use of XML with style sheet.

As we discussed, the basic rules of XML aren't too complicated. Although XML is admittedly more rigid than HTML. It's this consistency in structure that makes XML such as useful technology in representing diverse data.

Schemas and SML data modeling

What is Schema. The rigidity of XML as a markup language will no doubt make systems more robust. The facet of XML that allows errors to detected is schema, which is a construct that allows developers to define the format and structure of XML data.

When we create our own markup language, we are basically establishing which elements (tags) and attributes are used to create documents in that language. Although we could create genealogical XML documents using the elements and attributes, there really needs to be set of rules somewhere that establishes the format and structure of document created in the language. This set of rules is known as the SCHEMA for a markup language. A schema describes the particular class of data, you can create structured XML documents that adhere to the model. The real importance of schema is that they allow XML documents to be validated

for accuracy. This means that a schema allows an XML developer to process a document and see if it adheres to the set of constraints laid out in the schema.

To help clarify the role schemas play in XML, let's deal with an example.

my email id Gandra_mb*yahoo.com.
There's something wrong with it, yes you are right!!

As you guessed my email is Gandra_mb@yahoo.com . The differnce between these two emails is the symbols "*" and "@". The domain name@ is a fixed structure/symbol).

This is indeed, a simple schema….. we used a this schema to "validate' this email address and determine that is an error. The fix is to replace the * with @. I think, It solves the mystery of schema.

As we understand, the main reason schemas are used in XML is to allow machine validation of document structure. In the invalid e-mail example, we were easily able to see a problem because we knew that email address couldn't have * . But how would an email application be able to make this determination? The developer of the application would have to write specific code to make sure that

email addresses are structured to follow a given syntax, such as the name and domain name being separated by an "@" symbol. Like, application developer above, an XML document creator uses a schema. This schema can then be used by XML applications to ensure that documents are valid schemas provide a mechanism to automate the process of validating XML documents.

When it comes to creating schemas, there are two different approaches we can consider; Document Type Definition(DTD's), XML Schemas.

Let's start look at the description of document type definitions;
DTD's represent the original approach of creating a schema for XML documents. DTD's made it into XML because it eased the transition from SGML to XML- many SGML tools existed that could be used for XML. Now there is a more powerful approach to establishing schemas than DTD.'s. Even today, some use DTD's as standard schema technology for XML.

The main drawback to DTDs' is that they are based upon a somewhat cryptic language. When XML gives a good structural approach format data, why to concentrate on DTDs.

Example of DTD For XML document

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

<!ELEMENT list (recipe+)>
<!ELEMENT recipe (recipe_name, author?,
  meal?, ingredients, directions)>
<!ELEMENT recipe_name (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT meal (#PCDATA|course)*>
<!ELEMENT course (#PCDATA)>
<!ELEMENT ingredients (item+)>
<!ELEMENT item (#PCDATA)>
<!ATTLIST item
        measurement CDATA #IMPLIED
        unit       CDATA #IMPLIED>
<!ELEMENT directions (step*)>
<!ELEMENT step (#PCDATA)>
```

if we look at the above code, we can see ELEMENT precedes each element that can be used in XML document. Also the attributes for the recipe elements are listed after the word ATTLIST , in this case there is only one attribute are listed out beside attribute measurement. Although there are few strange looking piece of information in this DTD, such as <! At the beginning of each line and (#PCDATA) following each element, it's apparent that DTD'S aren't' too complex.

With the good exposure to DTD'S let's go to the better exposure of XML Schemaexact elements and attributes that are available within a given markup language, along with which attributes are associated with which elements and the relationships between the elements.

The process of creating a schema for an XML document is known as data modeling. Because it involves a class of data into elements and attributes that can be used to describe the data in an XML document. Once a data model(schema) is in place for a

XML schema

XML Schema is a new technology that is designed to replace DTDS' with amore powerful approach to creat schemas for XML based markup languages.

It is a collection of semantic validation rules designed to constrain XML data values. As an alternative to DTD, schemas are growing popular for the two reasons

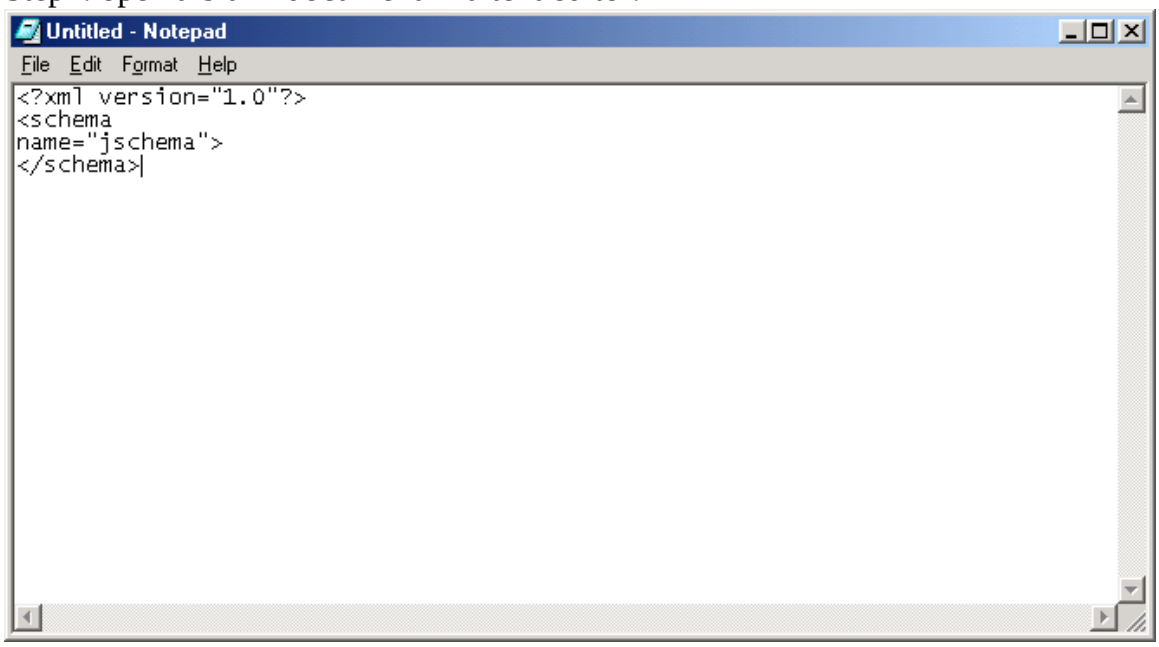
Unlike, DTD'S schemas are implemented in standard XML syntax.
Also unlike DTD'S schemas allow us to validate sophisticated data types like integer, date and time.

Syntax for creating Schema:
<Schema
name= "schema name"
xmlns= "urn:schemas-microsoft-com:xml-data"

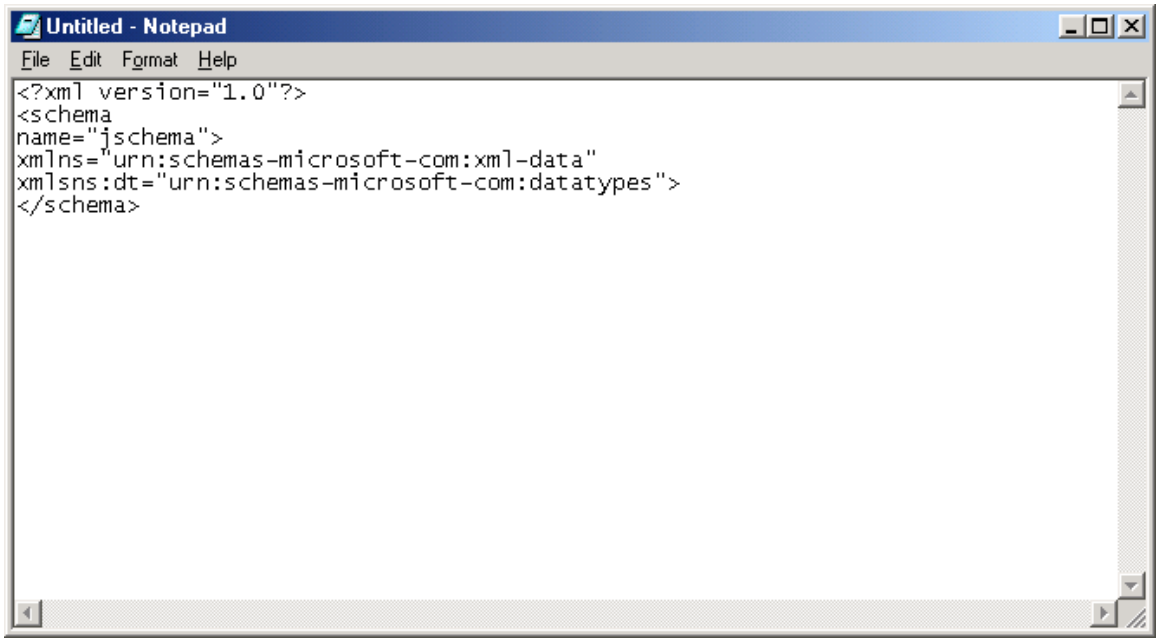
xmlns:dt= “urn:schemas-microsoft-com:datatypes”>

We are considering Microsoft’s MSXML because it offers the most advance support for schemas.

Let’s get ready to make a schema by ourselves by following steps below;
Step1. open a blank document in a text editor.



Type in the beginning and ending <schema. Tags after XML declaration in the very beginning. And input the name for schema as “jschema”. This is the name of the schema.

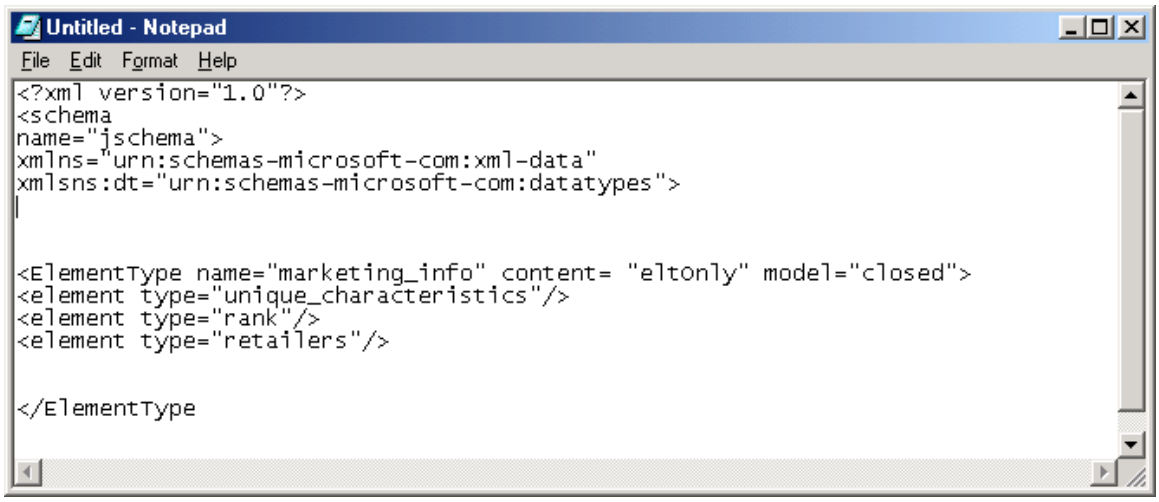


In the picture above, we typed the value for the xmlns attribute of <schema> tag, which includes Microsoft's schema resource as an implicit namespace. Doing so will allows to reference the special elements we need to create a schema, including element type, element , attribute type and attribute. . here the attribute is xmlns:dt. We are also including the urn:schmeas-microsoft-com:datatypes an an explicit namespace. Which reference to Microsoft's built-in data types, including string, integer and Boolean.

Adding elements to schema.

We can input contained relationships using schema. Container elements sometimes are referred to as aggregate elements because they define no original values; instead , they consist fo one or more other, predefined elements. Here's the syntax required to define an XML element as a container element.

<ElementType name= "conatainerElementName"
content = "eltOnly" model= "closed">
<element type= "containedElementName1">
<element type= "containedElementName2">
<element type= "containedElementNameN">
</ElementType>



Description of syntax with example: After including the xml declaration and schema syntax type in the beginning and ending<ElementType> tags.
(a)Type in a value for the name attribute of <ElementType> in this example it is marketing_info
(b)Type in the eltOnly value for the content attribute of the <ElementType>
(c)Specify one or more elements you want to be contained, using <element> tag
in this example, the unique_characteristics, rank and retailers elements are defined as contained elements.
(d)Type in a value of "closed" for the model attribute of the ElementType tag.

These types can be classified according the kind of data they represent. Following are the major categories of simple data types supported in XSD language, along with the specific XSD elements associated with each category
String types - xsd:string
Boolean types- xsd:Boolean
Number types-xsd:integer,xsd:decimal,xsd:float, xsd:double

Let's consider the above example again and see these model types
<xsd:element name= "name" type= "xsd:string"/>
<xsd:element name = "title" type= "xsd:string"/>
<xsd:element name = "occupation" type= "xsd:string"/>

We have successfully created the schema with including the element types. We can also input external elements, onetime elements, optional and repeated elements. This is the prerunner example for all other schema building techniques. We would be discussing several different types of data types that make it possible to model a wide range of data in XML documents..

Extensible Style language(XSL) fundamentals

We have discussed in earlier sections about styling the xml document, probably structuring.
As we mentioned let's dig into another prominent styling extensible style language fundamentals.

XSL is amore complex style sheet technology than cascading style sheet. In fact, XML is designed to do a lot more than just formal XML content for display purposes; it creates an ability to completely transform XML documents.

What is a style sheet/ Style sheets are special documents or pieces of code that used to format XML content for display purposes. This definition is perfect for CSS, which is a style sheet technology that originated as a means of adding good control of HTML content formatting. XSL is also style sheet technology, but it reaches beyond the simple formatting of content by also allowing to transform content. XSL goes further than CSS in its support for manipulating the structure of XML documents.

It is important to understand how an XSL style sheet isproceesed and applied to an XML document. This tak begins with an XML processor , which would be responsible for reading an XML document and processing it into meaningful pieces of information known as nodes. More specifically, every element and attribute in a document represents a node in the tree representaion of a document. The XSL processor starts with the root node in the tree and uses it as the basis for performing pattern matching in the style sheet. Pattermatching is the process of using patterns to identify nodes in the tree that are to be processed accord to XSL style sheets. The XSL processor analyzes templates and the patterns associated with them to process different parts of the document tree. When match is made, the portion of the tree matching the given pattern processed by the appropriate style sheet template.Let's start discussing XSL with an example

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
  <html>
```

```
 <body>
  <table border="2" bgcolor="yellow">
   <tr>
    <th>Name</th>
    <th>Course</th>
    <th>Registration</th>
    <th>Major</th>
    <th>Location</th>
   </tr>
   <xsl:for-each select=“compscifall/compsci">
   <tr>
    <td><xsl:value-of select="Name"/></td>
    <td><xsl:value-of select=“Course"/></td>
    <td><xsl:value-of select=“Registration"/></td>
    <td><xsl:value-of select=“Major"/></td>
    <td><xsl:value-of select="Location"/></td>
   </tr>
   </xsl:for-each>
  </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

The actual XML document for the above style sheet induced XML.

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="listing.xsl"?>
<ComSci>
< ComSci>
    <Name>Magnus</Name>
    <Unit>M.S</Unit>
    <Location>Bridgeport</Location>
    <semester>fall</semester>
    <School>UB</School>
</ ComSci>
< ComSci>
    <Name>John Mayers</Name>
```

```
    <Unit>B.S</Unit>
    <Location>Fairfield</Location>
    <semester>spring</semester>
    <School>offsite</School>
</ ComSci>
< ComSci>
    <Name>Robert</Name>
    <Unit>MBA</Unit>
    <Location>Wahlstrom</Location>
    <semester>winter</semester>
    <School>danna</School>
< ComSci> …..
</IComSci >
```

Discussing the example and syntax of XSL.

In order to understand the relelvance of XSL technologies, it's important to examine the role of the XSL processor once more, The XSL processor is responsible for performing two fundamental tasks:
1.construct a result tree from a transformation of a source document tree.
2.Interpret the result tree for formatting purposes.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
```

The template is an XSL construct that describes output to be generated based upon certain pattern matching criteria.The idea behind the template is to define a transformation mechanism that applies to certain portions of an XML document, which is a node or group of nodes.

Templates are defined in XSL style sheets using the xsl:template element, which is primarily container element for patters, expressions, and transformation logic. Container elements discussed in the previous documents.

The following template would come in useful for XML documents that have elements named xsl:template match="/"> for compscifall/compsci. As in the above example. Matching a portion wouldn't mean much if the tem[late didn't carry out any kind of transformation. Transformation logic is created using several template constructs that are used to control the application of templates in XSl style sheet. Let's move to the next description.
Xsl:value .of – inserts value of an element or attribute
Xsl:if- performs a conditional selection (please refer example)
Xsl:for .each-loops through the element in a document
Xsl:apply.templates-applies a template in a style sheet.

In the our XSL example above The xsl:value.of element requires an attribute named select that identifies the specific content to be inserted.
"Comspcifall/compsci"

```
<xsl:for-each select="compscifall/compsci">

 <td><xsl:value-of select="Name"/></td>
 <td><xsl:value-of select="Course"/></td>
<td><xsl:value-of select="Registration"/></td>
```

in the code above, the select attribute is set to compscifall/compsci.., which indicates that the current node is to be inserted into the result tree. The value of the select attribute works very much like the path of a file on a hard drive. This path indicates the folder hierarchy of the file. In similar way, the select attribute specifies the location of the note to be inserted in the result tree. A dot indicates a node in the current context, as determined by match attribute. An element or attrite name indicates a node beneath the current node, whereas two dots indicate the parent of the current node

Patterns and Expressions.

Patterns and expressions are used in XSL template technology to perform matches and are ultimately responsible for determining what portions of an XML document are passed through particular template for transformation. A pattern describes a branch of an XML tree, which is in turn consists of a set of hierarchical nodes. Patters are used throughout XSL to describe portions of a document tree for applying templates.

Expressions are similar to patterns in that they also impact which nodes are selected for transformation. However, expressions are capable of carrying out processing of their own, such as mathematical calculations, text processing and conditional tests. XSL includes numerous built in functions that are used to construct expressions with style sheet. Example

```
<td><xsl:value-of select="Major"/></td>
  <td><xsl:value-of select="Location"/></td>
```

This code demonstrates how to use the standard selection function to see major and location attributes. Although there is a fair amount of code in this XSL template , the functionality of the code is relatively very straightforward. The style sheet begins by declaring the XSL name space. An xsl .for.each element is used to loop through the "compsci" information. More specifically, templates are applied to name, major, location course and registration for child elements. Similar templates are defined for the remaining child elements, which are transformed and formatted in a similar fashion. The further discussion would be on XML documents, Xpath and Xlink.

XML Document Object Model(DOM)

What is the DOM?

The DOM is a standard method for exposing the elements in a structured document as data structure in our programming language of choice. A program, called DOM parser, reads through the XML in a file(or from some other source of input), and provides a data structure that can be accessed from the applications(front-end). In XML , DOM parser can also write its datastructure out as XML so that any changes that made to the data structure.

Like XML, DOM is a standard developed by the Worldwide web consortium. Most of the details how the DOM works are left up to the specific implementations. There are three different levels of DOM. Level1 is concerned with basic XML and HTML.

Level2 adds specifications for XML namespaces, cascading style sheets, events and various traversal schemes. Level3 will round out XML coverage, support for more user interface events and support Xpath(would be discussed in later parts). More DOM specifications are available @ http://www.w3.org/DOM/

DOM functionality

The DOM represents an XML document as a tree structure, where each element in the document is a node on the tree. Each node is associated with an interface that is more specific than the generic node interface. Some nodes are elements, other are attributes, and still others are comments. Each interface has its own properties and methods.

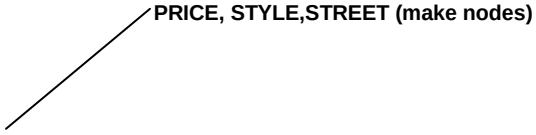
For example, the top-level node in the DOM model is a Document object. It can have exactly one Element object as child, one Document type object and one DOMImplementaion object. When dealing with XML, various interfaces supplied correspond to the structural entities in XML documents. If we deal with HTML, then the interfaces associated with them are also included in the DOM hierarchy.

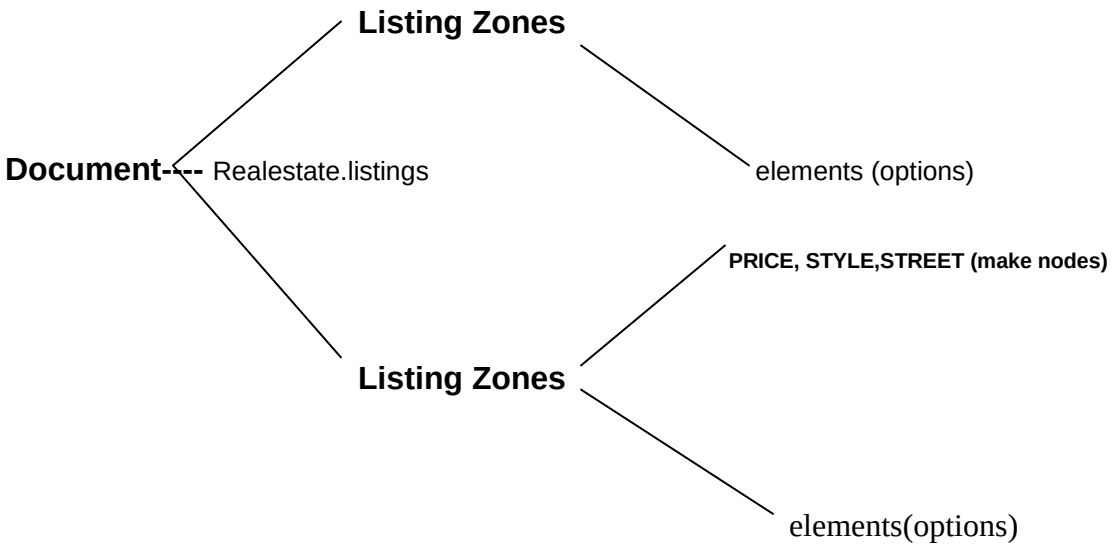
Let’s start with an example as we are dealing since the beginning for better understanding.

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

```
<REALESTATE.LISTINGS>
  <TITLE>Test of Real Estate data using XML and XSL</TITLE>
  <LISTING ZONED="Residential">
    <PRICE>99,000</PRICE>
    <STYLE>Victorian</STYLE>
    <STREET>113 Granger Road</STREET>
    <BEDROOMS>3</BEDROOMS>
  </LISTING>
  <LISTING ZONED="Residential">
    <PRICE>129,900</PRICE>
    <STYLE>Contemporary</STYLE>
    <STREET>635 Darlington Road</STREET>
    <BEDROOMS>4</BEDROOMS>
  </LISTING>
  <LISTING ZONED="Commercial">
    <PRICE>55,000</PRICE>
    <STYLE>Storefront</STYLE>
    <STREET>35 Main Street</STREET>
    <BEDROOMS>0</BEDROOMS>
  </LISTING>
    </REALESTATE.LISTINGS>
```

DOM Representation of an XML document.





Drawing is the DOM representation of the document mirrors the structure of the XML document, which shows the subsets of the interfaces available in the DOM. Realestate node expressed the element interface and the make nodes express the attribute interface.

DOM interfaces

Each of the interfaces is associated with a particular type of node found in a DOM tree. Rather than listing and describing all of these interfaces,i.e the interfaces frequently occur in the applications that use DOM.

The node interface

The node interface is the interface from which all other interfaces are derived.Regardless of what else a particular entity in a DOM tree.it's not a node still. The node interface exposes some attributes and methods that are generc to anything that's in DOM tree. These attributes and methods are largely associated with keeping track of the parents, siblings and children of the node. Nodes also have attributes that contain their names, values and a pointer to the document with which they are associated.

Document interface.

Document interface is the root node of DOM tree. There's one important thing to point here: Document node in DOM tree is not the root element of the XML document-it's one level above that. Every document has one child Element node that contains the root element of the XML document. Document nodes have other attributes that are associated with the document itself, rather than with the too element. That's why DOM tree has one more level in its hierarchy than an XML document.

In addition to the root level element of the document, the document node also contains a pointer to a document type node and DOMImplementation node. Neither of these nodes are commonly referenced on programs which use the DOM.

The Element interface

The element interface represents an element in an XML document. The only attributes specific to the Element node is the tag name. It includes methods that enable us to retrieve , add and remove attributes from the element. It also enables to retrieve child elements that have a specific name.

The Attr interface.

The Attr interface represents an attribute of an Element.Despite the fact thay they inherit the Node interface, they are not actually nodes on the DOM tree because they are not children of an element. Rather, they are part of the element itself.

Attributes have three properties – name, a value and a Boolean flag indicating wheterh if not the attribute was explicitly included in the document that was parsed.

The Nodelist interface

The Nodelist interface is different from the other interfaces. It's not a pointer to an entity in a DOM tree, rather it's an abstract data structure that enables DOM implementation to handle collections of node. For example, if we call the method of the Element interface that returns all the children of an element with a particular name, the collection of child elements is returned a s a Node list. You can then iterate over the Node list and extract all of the element nodes from it. The only interface is must implement is one that returns items by index. Its only attribute is the size of the collection. Using the size and the method, which returns items, you can iterate over the members of the collection.

Accessing the DOM within Internet Explorer

Internet explorer has some XML related features that make it perfect for this job. The main thing it has is a built-in DOM parser for XML. Turning an XML data structure into a DOM- based datastructure , this is due to the feature called XML data island, where explorer will automatically parse that XML document..

There are number of ways to retrieve the XML document object. Internet explorer provides an interface to every element in the document with an identifies. The following statement will assign the DOM representation of the XML data to a variable

Var xmldocument= document.all("name").XML Document

Let's break this statement into parts. The part of the statement to the lef of the equl sign initiates a new variable, called xmlDocument. Using var indicates that the variable is local, which just limits the scope of the variable- the parts of the page from which the variable is accessible.document.all("name") is used to retrieve the node in the HTML document with the ID name. The all property of the document object provides access to all of the named elements in a page. The XMLDocument property of the element with the ID name is the DOM representation of that element. This works because that element happens to be an XML data island.

Let's look at the a scrip that used to insert into document into the listing to print out the names of the nodes in the document tree using a function name printNode().

```
<script language = "JavaScript">
var xmlDocument= document.all( "name").XMLDocument;
printNode(xmldocument);
function printNode(node)
{
document.write('Node name;" + node.nodeName +  ,<br/>\n");

for (var I=0; I<node.ChildNodes.length; I++)

{
printNode (node.childNodes.item(i)0;
}
}
</script>
```

Let's look at this script in detail. The following code will be executed immediately;

```
Var xmlDocument = document.all( "name"). XMLDocument;
PrintNode(xmlDocument);
```

The rest of the code inside the script tag is inside the printNode function, and will be executed when the printNode function is called. The first line of the script creates anew variable, called xmlDocument, and assigns the XML document node from the XML that was included to the variable. The printnode(0 function will work with any node, including a document node, so as when we call it and pass it in the xmlDocument variable as argument.

UPDATING The DOM TREE

Not only does the DOM enables to access an XML data structure in a number of ways, it also enables to alter an XML data structure. Rather than providing you with a lengthy example program that explains how to alter the DOM tree. The key method here is the item() method of the NodeList interface. When you call item() method and supply it with an index, it returns the node associated with that index. To access the root level element of the document, the following code would be used.

Doc.chidlNodes.item(0). In order to be valid XML, only one element at the top level of the tree, so to add a node , it should be added to the root element. We can assign the root element to a variable like this:

Var root = doc.ChildNodes.firstchild.
Now let's have a look at the method of updating and removing elements themselves . There are two methods to do so—to replace child nodes or remove them, or to create a new element to put in place so the old one and then call the method that swaps them.

Var replacementElem=doc.Createelement("name")
Var oldNode=root.replaceChild(replacementElem, root.Childnodes.lastChild);

So the node was replaced is assigned to the variable oldNode.We have discussed the method of updating the database using the node logic.

I would like to provide an overview of subjects we've deal in descriptive as an aerial to make it in capsule/recap.

Review

There is a lot of hype surrounding XML, and a lot of hype surrounding Java. Together these technologies propose to solve many of the most common (and persistent) general computing problems that have been around for the last 20 years. XML and Java are not revolutionary in the approach to solving these problems of interoperability of code and data across and within platform and application boundaries. Rather, XML and Java provide solutions to these problems by using the most successful strategies and techniques that have been honed and refined over the last 20 years of
computing.

In the following paragraphs, I will highlight some of the most basic and important advantages that XML and Java provide to almost any system that uses them properly. This is by no means a comprehensive list of benefits, but items in this list should appear across just about any use of XML and Java technologies.

I will take a break from my normal pragmatic approach to getting you (the programmer) started with using XML and Java and just talk about the high level (design level) benefits of this wonderful combination. A good design is important to a good implementation for any system.

XML is structured

When you create your data using an XML editor (that you can write), you can not only input the content of your data, but also define the structural relationships that exist inside your data. By allowing you to define your own tags and create the proper structural relationships in your information (with a DTD), you can use any XML parser to check the validity and integrity of the data stored in your XML documents. This makes it very easy to validate the structure and content of your information when you use XML. Without XML, you could also provide this validation feature at the expense of developing the code to this yourself. XML is a great time saver because most of the features that are available in XML are used by most programmers when working on most projects.

XML is platform independent!

Information in an XML document is stored in plain-text. This might seem like a restriction if were thinking of embedding binary information in an XML document. There are several advantages to keeping things plain text. First, it is easy to write parsers and all other XML enabling technology on different platforms. Second, it makes everything very interoperable by staying with the lowest common denominator approach. This is the whole reason the web is so successful despite all its flaws.

By accepting and sending information in plain text format, programs running on disparate platforms can communicate with each other. This also makes it easy to integrate new programs on top of older ones (without rewriting the old programs), by simply making the interface between the new and old program use XML. For example, if you have an address book document stored in an XML file, created on a Mac, that you would like to share with someone who has a PC, you can simply email them the plain text address book XML document. This cant be done with binary encoded information which is totally platform (and program) dependent.

Another example is web enabling legacy systems. It is very feasible to create a Java web ennoblement application server that simply uses the services provided by the underlying legacy system. Instead of rewriting the legacy system, if the system can be made to communicate results and parameters through XML, the new and old system can work together without throwing away a company's investment in the legacy system.

XML is an open standard

By making the W3C the keeper of the XML standard, it ensures that no one vendor should be able to cause interoperability problems to occur between systems that use the open standard. This should be reassuring to most companies making an investment in this technology, by being vendor neutral, this solution proposes to keep even small companies out of reach of big companies choosing to change the standards on them. For example, if a big company chooses to change the platform at its whim, then most other companies relying on that platform suffer. By keeping all data in XML and using XML in communications protocols, companies can maximize the lifetime of their investment in their products and solutions.

XML is language independent

By being language independent, XML bypasses the requirement to have a standard binary encoding or storage format. Language independence also fosters immense interoperability amongst heterogeneous systems. It is also good for future compatibility. For example, if in the future a product needs to be changed in order to deal with a new computing paradigm or network protocol, by keeping XML flowing through the system, addition of a new layer to deal with this change is feasible.

DOM and SAX are open, language-independent set of interfaces

By defining a set of programming language independent interfaces that allow the accessing and mutation of XML documents, the W3C made it easier for

programmers to deal with XML. Not only does XML address the need for a standard information encoding and storage format, it also allows programmers a standard way to use that information. SAX is a very low level API, but it is more than what has been available before it. DOM is a higher level API that even provides a default object model for all XML documents (saving time in creating one from scratch if you are using data is document data). SAX, DOM and XML are very developer friendly because developers are going to decide whether this technology will be adopted by the majority and become a successful effort towards the goal of interoperable, platform, and device independent computing.

XML is web enabled

XML is derived from SGML, and so was HTML. So in essence, the current infrastructure available today to deal with HTML content can be re-used to work with XML. This is a very big advantage towards delivering XML content using the software and networking infrastructure already in place today. This should be a big plus in considering XML for use in any of your projects, because XML naturally lends itself to being used over the web.
Even if clients don't support XML natively, it is not a big hindrance. In fact, Java with Servlets (on the server side) can convert XML with style sheets to generate plain HTML that can be displayed in all web browsers. Using XML to pass parameters and return values on servers makes it very easy to allow these servers to be web-enabled. A thin server side Java layer might be added that interacts with web browsers using HTML and translates the requests and responses from the client into XML, that is then fed into the server.

XML is extensible

By not predefining any tags in the XML Recommendation, the W3C allowed developers full control over customizing their data as they see fit. This makes XML very attractive to encoding data that already exists in legacy databases (by using database metadata, and other schema information). This extensibility of XML makes it such a great fit when trying to get different systems to work with each other.

XML supports shareable structure

Since the structure of the XML document can be specified in DTDs they provide a simple way to make it easier to exchange XML documents that conform to a DTD. For example, if two software systems need to exchange information, then if both

of the systems conform to one DTD, the two systems can process information from each other. DTDs are not as powerful as some kind of schema architecture for XML, they don't support typing, sub classing, or instantiation mechanisms that a schema architecture must have.
DTDs are a simple way to make sure that 2 or more XML documents are of the same "type". It's a very limited approach to making "typed" XML documents shareable across systems. In the future some kind of schema system will be proposed by the W3C that should allow typing, instantiation and inheritance of information (in XML).

All of the advantages of XML outlined so far all make interoperability possible. This is one of the most important requirements for XML, to enable disparate systems to be able to share information easily. By taking the lowest common denominator approach, by being web enabled, protocol independent, network independent, platform independent and extensible, XML makes it possible for new systems and old systems (that are all different) to communicate with each other. Encoding information in plain text with tags is better than using proprietary and platform dependent binary formats.

Benefits of using XML

By using XML and Java, you can quickly create and use information that is properly structured and valid. By using (or creating) DTDs and storing your information in XML documents, you have a cross-platform and language independent data validation mechanism (for free) in all your projects!
You might use XML to define file formats to store information that is generated and used by your applications. This is another use of the structured nature of XML. The only limitation is that binary information can't be embedded in the body of XML documents. For example, if you wrote a word processor in Java, you might choose to save your word processor documents to an XML (actually your Application) file. If you use a DTD then your word processor would also get input file format validation as a feature for free. There are many other advantages to

using XML and a file storage format for your applications which will be illustrated later in the chapter.

Here are some benefits of the structured nature of XML:

- XML parsers make your application code more reliable and quick to develop by providing validity checking on your XML documents (if you use a DTD).
- XML allows you to easily generate XML documents (that contain your information), since it is so structured.
- XML parsers allow you to code faster by giving you a parser for your all your XML documents (with and without DTDs).

- Makes developers more productive by Simplifying the tight integration of XML with database data

- Extending the power of database queries to XML documents Makes applications more scalable by Processing database data and XML together in the same server, eliminating network traffic for data access.

- Makes applications more cost-effective to deploy by Providing a reliable, scalable, and manageable server platform supporting industry standards

- Running on your current and future hardware from NT & Linux boxes to parallel clusters of mainframe-class machines

XML documents may be stored in files or databases. When stored in files, XML documents are simply plain text files with tags (and possibly DTDs). It is very easy to save your XML documents to a text file and pass the text file around to other machines, platforms and programs (as long as they can understand the data). In the worst case scenario, XML documents (files) can be viewed in a text editor on just about any platform. XML documents are also naturally committed to a database (relational or object) or any other kind of XML document store.

Appendix

XML on the Web

Quick Reference
http://www.mulberrytech.com/quickref/index.html

Official Site
http://www.w3schools.com/

IBM site -- great training, news
http://www.ibm.com/developer/xml/

Webmonkey (Covers XML, XSLT)
http://www.w3.org/XML/http://hotwired.lycos.com/webmonkey/authoring/xml/index.html

From O'Reilly & Associateshttp://www.xml.com/

Scientific American: XML and the Second-Generation Webhttp://www.sciam.com/1999/0599issue/0599bosak.html

Microsoft Developer Network, XML Resourceshttp://msdn.microsoft.com/xml/

WML (Wireless Markup Language) Tutorialhttp://www.wirelessdevnet.com/training/WAP/WML.html

BOOK REFERENCES:

The XML handbook
Authors: Charles f. Goldfarb
Paul prescod

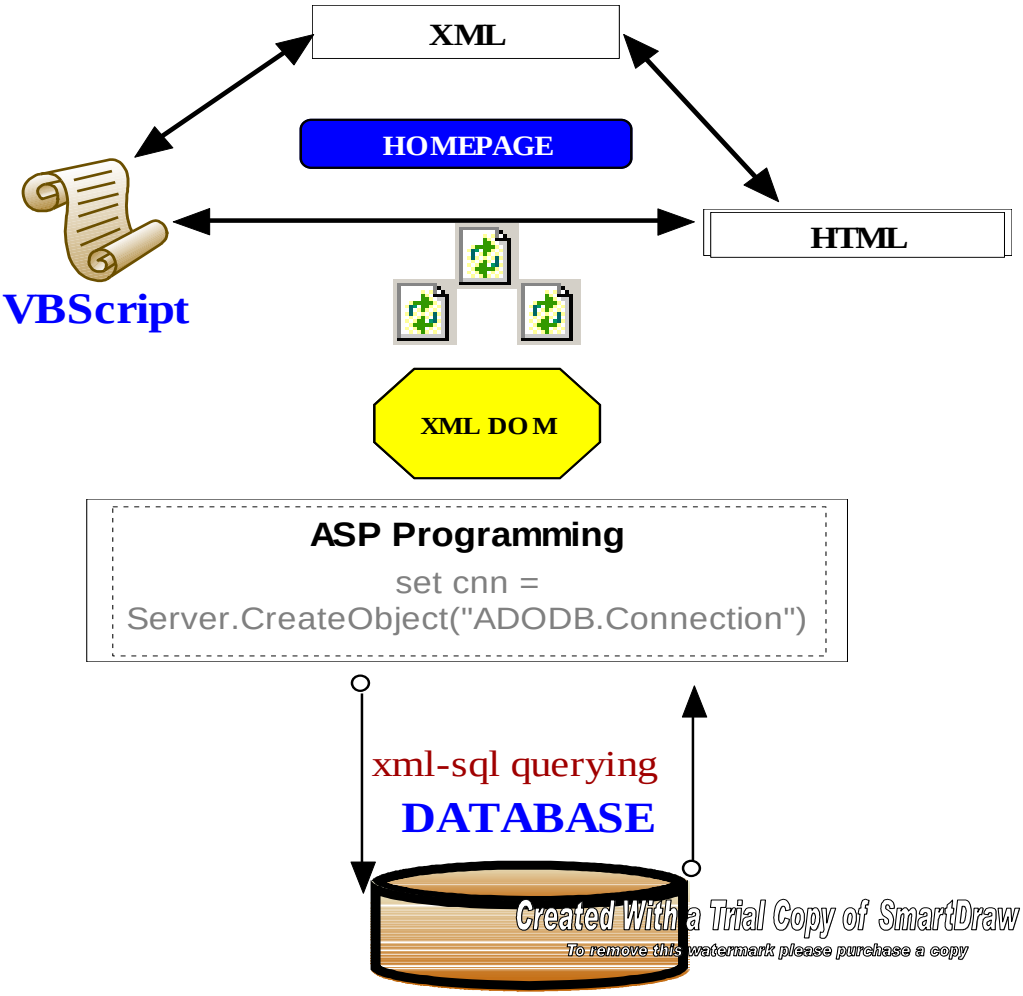
Xml Bible
Author: Elliotte Rusty Harold

XML in a Nutshell
Authors: Elliotte Rusty Harold
W. Scott Means

XML Schema
Author:Eric Van Der Vlist

Thank you very much!

XML APPLICATION OVERVIEW



SCHEMATIC REPRESENTATION OF THE APPLICATION

(Smart Draw software is used to draw the pictorial representation above)

Application is briefly described below to give an overall view with the description of Each functionality step by step.

INTRODUCTION:

This is an online data-tracking tool, which takes inputs from the user. The Front End is designed (HTML) and includes XML Data islands. This application also has modified, delete and clear data directly from the homepage (Front-End). I have used an Access database as a back-end database.

Since this application has been developed by me very recently, until the number of user and data storage increases (very soon), I would be requesting for tables in ORACLE. So I have included the option to switch between SQL and access databases.

Detailed description with each every command is dealt in the coding part of the thesis. (Please refer powerpoints & coding section)

Homepage Development:

I have developed an HTML page with the tables and frames (please refer homepage). And called JavaScript to highlight the selected records. Whenever the mouse rolls-over the data, the corresponding record will be highlighted. The main functionally of this application is DATAISLANDS. User would be able to see the Total Backend on the homepage through the Front-end. So that User has a comprehensive exposure to all records stored in the backend.

This has been achieved using XML & Visual Basic interface. Commands like xmldso.recordset.AbsolutePosition to interact with XML data islands created by Microsoft DOM call-ups.

The functionalities like UPDATE, INSERT, DELETE & CLEAR would be discussed extensively below.

Dealing with RECORDS

Before creating any file/function, connection files should be configured, Where the connection to the database is established with set cnn = Server.CreateObject ("ADODB.Connection"). And xml function file which creates the data islands for the existing records (access data base) is created with the function Set objNode = objDom.createElement (strName), which determined the various nodes starting from child node with appending functionality. Each and every command is clearly described in the coding section of the application. And also each value typed in from the front is validated by arCpt(GC_Dos) = oNode.dos.nodeTypedValue. Similarly all XML function has been created (Available in XML func.asp file).

Schema has been decided to establish the connection between the front-end and backend. Two layers of xml functionality, one between front and backend, another one is an interface between connection asp file and access database.

Note, in the all following files, xmlfunction.fle and conn.file should be used as include files.

Every file is described separately with basic simple steps.

INSERT.asp

Connection has been configured as we discussed above.

6. Data has been extracted from the values from the incoming XML (front –end interface) with command strCustomerID= GetNodeValue (objDOM, "/Supplier/Supplier ID").
7. Building the Update SQL statement strSQL = strSQL & "Insert Into Supplier". similar queries have been used to call the remaing values.
8. Finally the queries are executed with cnnNW.Execute strSQL
9. After executing the query, the main would be directed to confirmation Page.
10.

MODIFY. asp

6. Set objDOM = Server.CreateObject ("Microsoft.XMLDOM") is used to Create an XML DOM from the incoming XML. DOM function is described in the earlier part of the thesis.
7. 'Extract the values from the incoming XML object, strCustomerID = GetNodeValue (objDOM, "/Supplier/Supplier ID")

8. Building the SQL update Query with the command strSQL = strSQL & " Update Supplier Set ", similar for remaining records.

9. Executing the updated Sql query cnnNW.Execute strSQL.

DELETE.asp

1.Like the other folders step1, Creating an XML DOM from the incoming XML set objDOM = Server.CreateObject("Microsoft.XMLDOM") objDOM. Load(Request)

2. Defining the Delete SQL statement: strCustomerID= GetNodeValue (objDOM, "/Supplier/Supplier ID") for eliminating the respecting fields selected in the front –end.

3.Executing the Sql query established above to remove the data from database.

Discussing about XML process every time the above function is executed.

Each function is breakdown in simple steps as below:

1. Set the Connection to the database using Dim cnnXML, strXML, objXML, objXSL.

2. Obtain the XML data from the specified (supplier ID) database using strXML = GetXMLData (cnnXML, '').

3. Creating an instance of the XML documents set objXML = Server.CreateObject ("Microsoft.XMLDOM").

10. Merging XML files with XLT files.
Using the command Set objXSL = Server.CreateObject ("Microsoft.XMLDOM")

6. Loading the XML object with XML from the database using ObjXML.async = false ObjXML.loadXML strXML

7. Loading the XSL from the XSL file using objXSL.async = false and ObjXSL.load Server.MapPath ("NWDetails.xsl")

Sending Data to the front end on request basis

1.Initiating the XML transferring functionality at the beginning with objXML.transformNodeToObject objXSL, Response.

2.Determine the system functions from the previously called XMLfunction file with Function GetXMLData (cnnXML, strDataType) and dim rsXML, strXML, strSQL, stStream; strSupplier ID, DQ, Since the Primary key of the database table is Supplier ID. Such as strSupplier ID = Request ("cID").

3. Using the Loop to send every time the data is requested by the user.

```
Do While Not rsXML.EOF
Loop
strXML = strXML & "</root>"
```

4.Supplying the Front end tables with data such as GetXMLData = strip (strXML)

For elaborate explanation of each and every single function please refer the program files section.

Conclusion:

This application proves XML as the excellent backend technology for storing and sharing data in a highly structured manner. And also demonstrates XML provides the framework for creating customized markup languages, as we have customized for our data needs in the application discussed above. So the data would have the flexibility of spreadsheet along with the visual accessibility of a web page. The XML files are platform and browser independent, So Irrespective of the platform standard the above function would work with same setups and behavior of data modeling.

Extensive study on XML is presented in the study part of this thesis.