KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore-21

SUBJECT NAME: XML **SUB. CODE: 17CAP404W**
SEMESTER: IV **CLASS : II MCA**

Scope:
XML is a universal standard for information markup that is revolutionizing the way of handling data. This course will help students to learn how to create websites using XML, understand the essentials of the XML standards and appreciate some of the performance characteristics, gain some practical experience and insight of using XML.

Objective: To help students to

- Be aware of a range of XML tools (Many of them are free).
- Know how to set out an XML document
- Define custom markup language
- Understand the purpose of using DTDs and Schemas to validate XML
- Use XSLT to write a style sheet for XML document to produce multiple output
- Combine XML with existing web technologies.

UNIT I

World Wide Web: Introduction to the Internet and World Wide Web: W3C – History of Internet- World Wide Web- SGML – XML Resources – Internet and World Wide Web resources. Creating Mark up with XML: Introduction – Parsers and well formed XML Documents – Parsing an XML Document - Characters – Mark up – CDATA Sections – XML Namespaces

UNIT II

Document Type Definition – Parsers, Well formed and valid XML documents – Element type declarations – Attribute declarations- Attributes Types. Schemas:– Schemas VS DTD's – MSXML Schemas, W3C XML Schema –Declaring Types and Elements –Attribute constraints and Defaults - Simple type - Empty elements -Mixed content elements – Creating Attribute Groups

UNIT III

Document Object Model: DOM implementations – DOM with JavaScript – Components- Creating nodes – Traversing the DOM. Simple API for XML: DOM vs SAX – SAX based Parsers.

UNIT IV

XML Path Language: Nodes – Location Paths; XSLT: Templates - Creating Elements and attributes – Iteration and Sorting – Conditional Processing – Copying Nodes – Combining style sheets – variables. XLink, XPointer, XInclude and XBase.

UNIT V

XML Technologies and Applications: XML Query Language – Directory Services Markup Language – Resources Definition Framework – XML topic Maps – Virtual Hyper Glossary – Channel Definition Format – Information and Content Exchange Protocol – Platform for Privacy preferences – XML Metadata Interchange.

SUGGESTED READINGS

1. Deitel&Deitel. (2008), XML How to Program . 1st Edition, Pearson Education, New Delhi.
2. Ann Novarro, Chuck white, Linda Burman. (2000), Mastering XML, 1st Edition, BPB Publi, New Delhi.
3. Steve Holzner,(2001), Inside XML, 1st Edition, TechMedia, New Delhi.

WEB SITES

1. en.wikipedia.org/wiki/XML
2. www.w3.org/XML/
3. www.w3schools.com/xml/default.asp
4. http://www.cs.nmsu.edu/~epontell/courses/XML/material/xmlparsers.html#shapes_SAX

Continuous Internal Assessment **End Semester Examination –MarksAllocation**

S.No	Category	Marks
1	Attendance	5
2	Seminar	15
3	CIA I	10
4	CIA II	10
Total Marks		**40**

1	**Part A** 20 X 1 = 20 Online Examination	20
2	**Part B** 5 X 6 = 30 Either '**A**' OR '**B**' Choice	30
3	**Part C** 1 X 10= 10 Compulsory Question	10
	Total Marks	**60**

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Coimbatore-21

LECTURE PLAN

Staff name : Dr.S.Uma maheswari Subject code:17CAP404W
Subject Name : XML Class :II MCA
Semester : IV

S.No	Lecturer Duration(Hrs)	Topics to be Covered	Support Materials
UNIT - I			
1.	1	World Wide web :Introduction to the Internet and WWW- W3c	T1:1-3
2.	1	History of Internet- World Wide Web	T1:3-5, T1:6-7
3.	1	SGML-XML Resources	T1:6-7,W1
4.	1	Internet and World Wide web Resources	T1:19-20
5.	1	Creating markup with XML: Introduction- Parsers and well-formed XML Documents	T1:110-113,W2
6.	1	Parsing an XML documents- Character	T1:110-113,W2
7.	1	Mark Up- CDATA Section- XML Namespaces	T1:118—122 ,R1:59-62, T1:123-126, R1:82-84
8.	1	Recapitulation and Discussion of important Questions	
Total No. of Hours planned for Unit I			**8 hrs**

TEXT BOOK:

T1: Deitel & Deitel, 2008, "XML How to Program", 1st Edition, Pearson Education, New Delhi

REFERENCE BOOK:

R1: Steve Holzner , "Inside Xml", 1st Edition, Techmedia, New Delhi

WEBSITES:

W1: http://www.w3.org/MarkUp/SGML

W2: http://www.stylusstudio.com/xml/parser.html

W3:http://www.w3schools.com/xml

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Coimbatore-21

S.No	Lecturer Duration(Hrs)	Topics to be Covered	Support Materials
UNIT - II			
1.	1	DTD Parsers , Well formed and Valid XML documents	T1:134-136,W1
2.	1	Element Type declaration- Attribute Declaration	T1:145-147, R1:68-73
3.	1	Attribute Types, Schemas , Schema vs DTD's	T1:147-158,W2, T1:165-167,W3
4.	1	MS XML Schema, W3 XML Schema	T1:167-183,R1:200-204, T1:182-186, R1:204-208
5.	1	Declaring types and Elements Attributes constraints and defaults	T1:174-175
6.	1	Simple type, Empty Elements	T1:184-186
7.	1	Mixed Content Elements , Creating attribute Groups	T1:184-186, R1:215-224,W4
8.	1	Recapitulation and Discussion of important Questions	
Total No. of Hours planned for Unit II			**8 hrs**

TEXT BOOK:

T1: Deitel & Deitel, 2008, "XML How to Program", 1st Edition, Pearson Education, New Delhi

REFERENCE BOOK:

R1: Steve Holzner, "Inside Xml", 1st Edition, Techmedia, New Delhi

WEBSITES:

W1: http://www.w3schools.com>xml>xml-dtd

W2: http:// www.w3schools.com>xml>xml-dtd-attributes

W3:http://www.w3schools.com>xml>schema-simpleattributes.asp

W4: http://www.w3schools.com>xml>el-attributes-grow.asp

JOURNAL:

J1: "Controlling Accessing to XML documents", Volume 2 2001.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Coimbatore-21

S.No	Lecturer Duration(Hrs)	Topics to be Covered	Support Materials
UNIT - III			
1.	1	DOM , DOM Implementation	T1:192-194,W1
2.	1	DOM with Javascript, Components	T1:194-199 , R1:297-316, T1:199-209
3.	1	Usage of Components, Creating Nodew	T1:199-209, T1:209-213
4.	1	Traversing the DOM, Traversing the DOM with XML	T1:213-216
5.	1	Simple API for XML , Simple API for XML DOM	T1:232-234,W2
6.	1	Simple API for XML DOM vs SAX	T1:232-234,W2
7.	1	SAX based parsers	T1:234-235, R1:563-579
8.	1	Recapitulation and Discussion of important Questions	
Total No. of Hours planned for Unit III			**8 hrs**

TEXT BOOK:

T1: Deitel & Deitel, 2008, "XML How to Program", 1st Edition, Pearson Education, New Delhi

REFERENCE BOOK:

R1: Steve Holzner, "Inside Xml", 1st Edition, Techmedia, New Delhi

WEBSITES:

W1: http://www.w3.org>DOM

W2: sax.sourceforge.net

JOURNAL:

J1: "XML optimistic Concurrency Control for DOM API", Volume 1 July 2014.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Coimbatore-21

S.No	Lecturer Duration(Hrs)	Topics to be Covered	Support Materials
UNIT - IV			
2.	1	XML Path Language	T1:192-194,W1
2.	1	Nodes, Location Path	T1:194-199 , R1:297-316, T1:199-209
3.	1	XSLT , XSLT : Templates	T1:209-213
4.	1	Creating element and Attributes , Iteration and Sorting	T1:209-213, T1:213-216
5.	1	Conditional processing , Copying nodes	T1:232-234, W2
6.	1	Combining Style sheets - Variables	T1:234-235, R1:563-579
7.	1	XLink , XPointers, XInclude and XBase	T1:373-391,R1:721-751
8.	1	Recapitulation and Discussion of important Questions	
Total No. of Hours planned for Unit IV			**8 hrs**

TEXT BOOK:

T1: Deitel & Deitel, 2008, "XML How to Program", 1st Edition, Pearson Education, New Delhi

REFERENCE BOOK:

R1: Steve Holzner, "Inside Xml", 1st Edition, Techmedia, New Delhi

WEBSITES:

W1: http://www.w3.org>xpath

W2: http://www.w3schools.com>xml>xml-xpath

W3: http:// www.w3.org>xslt

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Coimbatore-21

S.No	Lecturer Duration(Hrs)	Topics to be Covered	Support Materials
UNIT - V			
1.		XML Technologies and Application XML Query Language	T1:601-603,W1
2		Directory Services Mark Up Language-Resources Definition Framework	T1:603-604, T1:604-608 ,W2
3		XML Topic Maps - Virtual Hyper Glossary – Channel Definition Format	T1:608-612 , T1:612-616,W3
4		Virtual Hyper Glossary – Channel Definition Format	T1:612-616,W3
5			
6		Information and Content Exchange Protocol	T1:616-617
7		Platform for privacy preferences	T1:617-618
8		XML Metadata Interchange	T1:620-626
9	1	Recapitulation and Discussion of important Questions	
Total No. of Hours planned for Unit V			**hrs**

TEXT BOOK:

T1: Deitel & Deitel, 2008, "XML How to Program", 1st Edition, Pearson Education, New Delhi

REFERENCE BOOK:

R1: Steve Holzner, "Inside Xml", 1st Edition, Techmedia, New Delhi

WEBSITES:

W1: http://www.w3.org>Standards>xml

W2: http://www.w3.org>xml>Query

W3: http://www.xml.coverpages.org>uhg

W4: http://searchsoa.techtarget.com/definition/XMI

UNIT-I

SYLLABUS

World Wide Web: Introduction to the Internet and World Wide Web: W3C – History of Internet- World Wide Web- SGML – XML Resources – Internet and World Wide Web resources. Creating Mark up with XML: Introduction – Parsers and well-formed XML Documents – Parsing an XML Document - Characters – Mark up – CDATA Sections – XML Namespaces

INTRODUCTION TO THE INTERNET AND WORLD WIDE WEB

INTRODUCTION

Welcome to the world of XML! We have worked hard to create what we hope will be an informative, entertaining and challenging learning experience for you. As you read this book, you may want to refer to our Web site. **www.deitel.com.** For updates and additional information on each subject.

The technologies you will learn in this book are intended for experienced professionals building substantial systems. Perhaps most important, the book uses working programs and shows the outputs produced when those programs are run on a computer. We present programming concepts in the context of complete working programs. We call this the live-code approach. These examples are available form three locations-they are on the CD- ROM inside the back cover of this book, they may be downloaded from our Web site www.deitel.com and they are available on our interactive CD-ROM product, the XML Multimedia Cyber Classroom. The Cyber Classroom's features and ordering information appear in the last few pages of this book, including short answers and small programs. If you purchased our boxed product The Complete XML Training Course, you already have the Cyber Classroom.

WORLD WIDE WEB CONSORTIUM (W3C)

In October 1994, Tim Berners-Lee founded an organization—called the World Wide Web Consortium (W3C)—devoted to developing non-proprietary, interoperable technologies for the World Wide Web. One of the W's primary goals is to make the Web universally accessible—regardless of disabilities, language, culture, etc.

The W3C is also a standardization organization. Web technologies standardized by the W3C are called Recommendations. Current W3C Recommendations include HyperText Markup Language (HTML), Cascading Style Sheets (CSS) and the Extensible Markup Language (XML). Recommendations are not actual software products, but documents that specify the role, syntax, rules, etc. of a technology. Before becoming a W3C Recommendation, a document primarily

asses through three major phases: Working Draft—which, as its name implies, specifies an evolving draft, candidate Recommendation—a stable version of the document that industry may begin implementing and Proposed Recommendation—a candidate Recommendation that is considered mature (i.e., has been implemented and tested over a period of time) and is ready to be considered for W3C Recommendation status. For detailed information about the W3C Recommendation track, see "6.2 The W3C Recommendation track" at

www.w3.org/consortium/process/process-19991111/process.html#RecsCR

The W3C is comprised of three hosts—the Massachusetts Institute of Technology(MIT), INRIA (Institute National de Recherché en Informatique at Automatique) and Keio University of Japan—and over 400 members, including Deitel & Associates, Inc. Members provide the primary financing for the W3C and help provide the Consortium. To learn more about theW3C visit **www.w3.org**

HISTORY OF THE INTERNET

In the late 1960s, one of the authors (HMD) was a graduate student at MIT. Is research at MIT's Project Mac (now the Laboratory for Computer Science—the home of the World Wide Web Consortium) was funded by ARPA—the Advanced Research Projects Agency of the Department of Defense. ARPA sponsored a conference at which several dozen ARPA-funded graduate students were brought together at the University of Illinois at Urbana-Champaign to meet and share ideas. During the conference, ARPA rolled out the blueprints for networking the main computer systems of about a dozen ARPA-funded universities and research-institutions. They were to be connected with communication lines operating at a then stunning 56 Kbps (i.e., 56,000 bits per second), at a time when most people (of the few who could be) were connecting over telephone lines to computers at a rate of 110 bits per second. HMD vividly recalls the excitement at the conference. Researchers at Harvard talked about communication with the Univac 1108 "supercomputer" across the country at the University of Utah to handle calculations related to their computer graphics research. Many other intriguing possibilities were raised. Academic research war about to take a giant leap forward. Shortly after this conference, ARPA proceeded to implement what quickly became called the Arpanet, the grandparent of today's Internet.

Things worked out differently than originally planned. Although the ARPAnet did enable researches to share each others' computers, its chief benefits proved to be the capability of quick and easy communication via what came to be known as electronic mail(e-mail). This is true even today on the Internet, with e-mail facilitating communications of all kinds among millions of people worldwide.

One of ARPA's primary goals for the network was to allow multiple users to send and receive information at the same communications paths (such as phone lines). The network operated with a technique called packet switching in which digital data was sent in small packages called packets. The packets contained data, address information, error-control information and sequencing information. The address information was used to route the packets of data to their destination, and the sequencing information was used to help reassemble the packets (which—because of complex routing mechanisms—could actually arrive out of order) into their original order for presentation to the recipient. This packet-switching technique greatly reduced transmission costs from those of dedicated communications lines.

The network was designed to operate without centralized control. This meant that if a portion of the network should fail, the remaining working portions would still be able to route packets form senders to receivers over alternates paths.

The protocols for communicating over the ARPAnet became known as TCP—the Transmission Control Protocol. TCP ensured that messages were properly routed form sender to receiver and that those messages arrived intact.

In parallel with early evolution of the Internet, organizations worldwide were implementing their own networks for both intro-organization (i.e., within the organization) and inter-organization (i.e., between organizations) communication. A huge variety of networking hardware and software appeared. One challenge was to get these to intercommunicate. ARPA accomplished this with the development of IP—the Internet Protocol),truly creating a "network of networks," the current architecture of the Internet. The combined set of protocols is now commonly called TCP/IP.

Initially, use of the Internet was limited to universities and research institutions; then the military become a big user. Eventually, the government decided to allow access to the Internet for commercial purposes. Initially there was resentment among the research and military communities—it was felt that response times would become poor as "the net" became saturated with so many users.

HISTORY OF THE WORLD WIDE WEB

The World Wide Web allows computer users to locate and view multimedia-based documents (i.e. documents with text, graphics, animations, audios and/or videos) on almost any subject. Even though the Internet was developed more than three decades ago, the introduction of the World Wide Web was relatively recent event. In 1989, Tim Berners-Lee of CERN (the European Laboratory for Particle Physics) began to develop a technology for sharing information by using hyperlinked text documents. He based his new language on the well-established

standard Generalized Markup Language (SGML)—a standard for business data interchange that we discuss in Section 1.6—and called it the HyperText Markup Language (HTML). He also wrote communication protocols to form the backbone of his new hypertext information system, which he termed the World Wide Web.

The Internet and the World Wide Web will surely be listed among the most important and profound creations of humankind. In the past, most computers applications ran on "stand-alone" computers, i.e., computers that were not connected on the. Today's applications can be written to communicate among the world's hundreds of millions of computers. The Internet mixes computing and communications technologies. It makes our work easier. It makes information instantly and conveniently accessible worldwide. It makes it possible for individual and small businesses to get world wide exposure. It is changing the nature of the way business is done. People can search for the best prices on virtually any product or service. Special-interest communities can stay in touch with one another. Researchers can be made instantly aware of the latest breakthroughs worldwide.

HISTORY OF SGML

In the late 1960's Charles Goldfarb, Edward Mosher and Raymond Lorie of IBM tackled the problem of building a powerful yet portable system for the interchange and manipulation of legal documents. At the time, communication between computer systems at IBM was hindered by a profusion of different file formats. The three researchers realized that communication would be best facilitated by a system-independent common format which was specific to legal documents. They decided to use a markup-language at the center of their system. Markup, which identifies structures in a document, was a mainstay of text processing and this compatible across many platforms.

The IBM team's prototype language marked up structural elements that specified the abstract nature rather than the formatting of the information they contained. The formatting information itself would be kept in separate files called style sheets, with which computers could format the elements and render a finished document. This method of structuring data made it possible for computers to automatically process documents in many new ways. Documents could only be processed reliably, the IBM team realized, if they adhered to a standard, so that a system would be able to recognize and reject invalid documents (i.e., document with missing information, extra information, etc.). The structure of each documents type, therefore was strictly defined in a file called document type definition (DTD). This separation of presentation and

validation provided great flexibility; because DTDs and style sheets could easily be modified without directly affecting he marked up data.

By 1969, the team of researchers had developed a language with all of these capabilities and called it the generalized Markup Language (GML). In 1974, Goldfarb proved that a parser (i.e., software capable of analyzing the structure and syntax of a document) could validate a document without actually processing it. This opened the door to further development, which culminated in the 1986 adoption of the Standard Generalized Markup Language (SGML) as international standard. If quickly became the business standard for data storage and interchange throughout the world. Processing of SGML documents is defined by the Document Style Semantics and Specification Language (DSSSL), another international standard. Six years after SGML became an international Organization for Standards (ISO) standard, it was followed by the Hypermedia/Time-based Structuring Language HYTIME), which was designed for SGML, representation of hypermedia and multimedia and has powerful linking capabilities.

XML RESOURCES

XRC, or XML Resource, or XML Based Resource System is a cross-platformXML-based user interface markup language used by wxWidgets. XRC allows graphical user interface elements, such as dialogs, menu bars and toolbars, to be stored as XML, which can be loaded into the application at run-time or translated into a target programming language and compiled.

CHARACTERS

In this section, we discuss the character set used in XML documents and some of its properties. A character set consists of the characters that may be represented in a document. For example, the ASCII (American Standard Code for Information Interchange) character set contains the letters of the English alphabet, the numbers 0–9 and punctuation characters, such as !, - and ?.

5.5.1 Character Set XML documents may contain the following characters: carriage returns, line feeds and Unicode® characters. Unicode is a standard of the Unicode consortium. Its goal is to enable computers to process the characters for most of the world's major languages. In Section 5.5.4, we demonstrate how to use Unicode in an XML document. Visit www.unicode.org for more information about the Unicode standard. 5.5.2 Characters vs. Markup Once a parser determines that all characters in a document are legal, it must differentiate between markup text and character data. Markup text is enclosed in angle brackets (< and >). Character data is the text between a start tag and an end tag. Child elements are considered markup—not character data. Lines 1, 3–4 and 6–8 in Fig. 5.1 contain markup text. In line 7, tags and are the markup text and the text Welcome to XML! is the character data. 5.5.3 White Space, Entity References and Built-in Entities Spaces, tabs, line feeds and carriage returns are characters

commonly called whitespace characters. An XML parser is required to pass all characters in a document, including whitespace characters, to the application using the XML document. An application may consider whitespace characters either significant (i.e., preserved by the application) or insignificant (i.e., not preserved by the application). Depending on the application, insignificant whitespace characters may be collapsed into a single whitespace character or even removed entirely. This process is called normalization. For example This is character data contains three significant whitespace characters in the character data. When this character data is normalized, the five spaces between character and data are collapsed into a single significant space.

Almost any character can be used in an XML document, but the characters ampersand (&), left-angle bracket (), apostrophe (') and double quote (") are reserved in XML and may not be used in character data. To use these characters in the content of an element or attribute we must use entity references, which begin with an ampersand (&) and end with a semicolon (;). Using entity references prevents XML processors from misinterpreting character data as XML markup. For example, angle brackets are reserved for delimiting markup tags. If angle brackets were found in a the content of an element or attribute, the XML parser would interpret these as XML markup and would incorrectly parse the document. In the next section, we demonstrate how to use entity references to represent Unicode characters in an XML document. The apostrophe and double quote characters are reserved for delimiting attribute values.

XML provides built-in entities for ampersand (&), left-angle bracket (<), right-angle bracket (>), apostrophe (') and quotation mark ("). For example, to mark up the characters "<>&" in element message we would write <>& Using these entities instead of the and & characters prevents the XML processor from mistaking these characters for XML markup.

The document begins with the XML declaration in line 1. Line 6 specifies the file—called a document type definition (DTD) file—that defines the structure (i.e., what elements the XML document must contain, what order the elements must have, etc.) of this document and the entities used by the document. This tag contains four items: the DOCTYPE, the name of the root element (welcome), the SYSTEM flag—which indicates that the DTD is located in an external file rather than the document itself and the name of the file (lang.dtd) containing the DTD.

MARK UP

This section elaborates on elements and their attributes as well as how they are used to construct proper XML markup. XML element markup consists of a start tag, content and an end tag. Unlike HTML, all XML start tags must have a corresponding end tag.

For example, is correct HTML, but in XML, the ending tag must also be supplied, as in

This type of element is called an empty element, because it does not contain content (i.e., character data). Alternatively, an empty tag may be written more concisely as which uses the forward slash (/) for termination. Elements define structure. An element may or may not contain content (i.e., child elements or character data). Attributes describe elements. An element may have zero, one or more attributes associated with it. Attributes are placed within the element's start tag. Attribute values are enclosed in quotes—either single or double. For example, element car

contains attribute doors, whose value is "4". XML element and attribute names can be of any length and may contain letters, digits, underscores, hyphens and periods; they must begin with a letter or an underscore.

CDATA Sections

In the previous section, we discussed markup. In this section, we discuss sections of an XML document—called CDATA sections—that may contain text, reserved characters (e.g., , ' and ". Figure 5.7 lists an XML document that compares text in a CDATA section to character data. The first sample element (lines 8–12) contains C++ code as character data. Each occurrence of and & must be replaced by an entity reference in order prevent syntax errors. Lines 15–20 use a CDATA section to indicate a block of text that the parser should not treat as character data or markup. CDATA sections begin with . Notice that and & characters (lines 18 and 19) do not need to be replaced by entity references. IE5 also preserves whitespace in the CDATA section, although this is not a requirement of XML processors.

XML Namespaces

Because XML allows document authors to create their own tags, naming collisions (i.e., two different elements that have the same name) can occur. For example, we may use the element book to mark up data about one of our publications. A stamp collector may also create an element book to mark up data about a book of stamps. If both of these elements were used in the same document there would be a naming collision and it would be difficult to determine which kind of data each element contained. Namespaces provide a means for document authors to prevent collisions.

INTERNET AND WORLD WIDE WEB RESOURCES

www.deitel.com

Please check this site for daily updates, corrections and additional resources for all Deitel & Associates, Inc. publications.

www.learnthenet.com/english/index.html

Learn the Net is Web site containing a complete overview of the Internet, the World Wide Web and the underlying technologies. The site contains information that can help people Internet and Web get novices started.

www.xml.com

This Web site contains information, resources, tutorials and links related to any aspect and application of XML.

www.xml.org

This Web site is the industry portal for XML and contains many useful resources.

www.oasis-open.org/cover

This web site is a portal to an extensive listing of articles and links concerning various XML technologies.

POSSIBLE QUESTIONS

PART B QUESTIONS
(EACH QUESTION CARRIES SIX MARKS)

1. Explain the history of the Internet and World Wide Web.
2. Explain the history of SGML.
3. Explain the process of 'W3C'.
4. Explain the Structure of XML with an Example.
5. Write a program to create a menu in XML.
6. Explain the features and components of XML with Illustration.
7. Explain the Process of XML Namespace.
8. Explain the function of CDATA Section.
9. Explain the history of WWW.
10. Explain the process of creating markup with XML.

PART C QUESTIONS
(EACH QUESTION CARRIES TEN MARKS)

1. Explain the process and with suitable examples i) XLink ii) XPointer
2. Explain about the DOM Traverse Node.
3. Write a program to storing XML Files on the server and explain in detail.
4. Write a java program to parsing an XML file using SAX.
5. Write a program to view an XML Student details and display the Student Details with an XSLT Stylesheet.

Karpagam Academy of Higher Education

Department of Computer Applications

Part-A Questions

Subject Code : 17CAP404W **Subject : XML** **Class : II MCA**

UNIT - I

S.No.	Questions	Choice 1	Choice 2	Choice 3	Choice 4		Answer
1	XML stands for	Extended Markup language	Extensible Markup Language	Extended Meaningful language	X-Mark Language		Extensible Markup Language
2	SGML stands for	Standard Geographic Marking Language	Simulated General Marking Language	Standard Generalized Markup Language	Simple& General Markup Language		Standard Generalized Markup Language
3	CSS stands for	Cascading Style Sheet	Common Standard style	Common Style Sheet	Cascade Structured Stylesheet		Cascading Style Sheet
4	DTD is an abbreviation of _____	Document Typing Device	Digital Typing Device	Document Type Definition	Direct Type Definition		Document Type Definition
5	DOM is an abbreviation of _____	Document Object Model	Document Of markup	Defining Object Model	Digital Object Model		Document Object Model
6	Document Style Semantics and Specification Language is abbreviated as	DSL	DSSL	DSSSL	DSSSSL		DSSSL
7	Hypermedia/ Time-based Structuring Language is abbreciated as	HTSL	HTL	HyTime	HMTSL		HyTime
8	_____ is a standards Organization	XML	Deitel Associations Inc.	ARPA	World Wide Web Consortium		World Wide Web Consortium
9	Which of these is not a W3C recommendation	XML	HTML	CSS	WWW		WWW
10	The technologies standardized by W3C are called _____	Standards	Recommendations	Languages	Markups		Recommendations
11	_____ are not actual software products, but documents that specifythe role, syntax, rules etc.,of a technology.	Standards	Recommendations	Languages	Markups		Recommendations
12	W3C Recommendations pass through _____ phases	2	4	5	3		3
13	_____ specifies an evolving phase of W3C Recommendation	Working Draft	Candidate Recommendation	Proposed Recommendation	XML		Working Draft
14	A stable version of the document - implemented by the industry is called _____	Working Draft	Candidate Recommendation	Proposed Recommendation	XML		Candidate Recommendation
15	A Candidate Recommendation that is considered Mature is called _____	Working Draft	Candidate Recommendation	Proposed Recommendation	XML		Proposed Recommendation
16	A Blueprint of networking a dozen of computer systems was proposed in a conference sponsered by	ARPA	W3C	MIT	INRIA		ARPA
17	The first network of computers is called	World wide web	Internet	ARPAnet	WebWorld		ARPAnet
18	The ARPAnet communication line operated at a rate of _____	100 bps	110 kbps	56 kbps	80 bps		56 kbps

19	Connecting of telephone lines to computers operated at rate of	100 bps	110 bps	56 kbps	80 bps		110 bps
20	_____ is the grandparent of today's internet	World wide web	Telephone network	ARPAnet	WebWorld		ARPAnet
21	The _____ on the header of each packet is used to find the destination to which the packet is sent	packet information	address information	sequence information	data information		address information
22	The ARPA network operated with a technique called	broadcasting	telephoning	data exchanging	packet switching		packet switching
23	Small package of digital data transmitted through the net is called _____	packets	data	nodes	files		packets
24	_____ is used to reassembling the received packets	packet information	address information	sequence information	data information		sequence information
25	The Protocols for communicating over the ARPAnet was known as _____	Transmission protocol	Transaction control Program	Transmission Connection Protocol	Transmission Control Protocol		Transmission Control Protocol
26	TCP stands for _____	Transmission protocol	Transaction control Program	Transmission Connection Protocol	Transmission Control Protocol		Transmission Control Protocol
27	The informaion carrying capacity of communication lines is called _____	bandwidth	communication protocol	information rate	net rate		bandwidth
28	_____ is the founder of W3C	Tim Berners Lee	Charles GoldFarb	Edward Mosher	Raymond Lorie		Tim Berners Lee
29	The Technology of sharing information using hyperlinked text document was developed by _____	Tim Berners Lee	Charles GoldFarb	Edward Mosher	Raymond Lorie		Tim Berners Lee
30	The formatting information kept in separate files called _____	XML Documents	Format Files	Style Sheets	Structuring Sheets		Style Sheets
31	The documents with missing or extra information are considered	incomplete documents	non-structured documents	invalid documents	unformed documents		invalid documents
32	The structure of each document is strictly defined in a file called	DOM	DTD	CSS	XSL		DTD
33	The team of researchers Charles GoldFarb, Edward Mosher and Raymond Lorie developed a language called _____	XML	SGML	GML	XSL		GML
34	The software capable of analyzing the structure and syntax of a document is called _____	compiler	translator	analyser	Parser		Parser
35	In 1974 Goldfarb proved that a _____ can validate a document without actually processing it.	compiler	translator	analyser	Parser		Parser
36	The first parser was developed by	Tim Berners Lee	Charles GoldFarb	Edward Mosher	Raymond Lorie		Charles GoldFarb
37	_____ is a subset of SGML	XML	HTML	XSL	DHTML		XML
38	_____ is an application of SGML	XML	HTML	XSL	DHTML		HTML

39	XML is a _____	Programming languge	Formatting language	Meta language	Processing Language		Meta language
40	TEI stands for	Text Enhancing Interface	Text Encoding Initiative	Transmission Enhancing Information	Encoding & Decoding		Text Encoding Initiative
41	_____ incorporates the elements of both CSS and DSSSL	XSL	XML	DSL	Xlink		XSL
42	XSL stands for	Extensible standard Languae	Extensible Stylesheet Language	Extended Simple Langage	XML		Extensible Stylesheet Language
43	____ combines the ideas from Hytime and TEI	XSL	XML	DSL	XLink		XLink
44	XLink in an acronym of	Extensible standard Languae	Extensible Stylesheet Language	Extended Library Langage	Extensible Linking Language		Extensible Linking Language
45	Who is known as the father of Internet	Tim Berners Lee	Charles GoldFarb	Edward Mosher	Raymond Lorie		Tim Berners Lee
46	XML files are stored in _____ format	image	text	binary	ASCII		text
47	XML files names commonly end with the extension _____	xsl	txt	bmp	xml		xml
48	Version of the XML used is mentioned in _____	comment statement	root element	xml declaration tag	any user defined tag		xml declaration tag
49	If the version of XML is not specified it is assumed to conform to _____ version.	latest	1.0	1.2	2.0		latest
50	All XML documents must contain _____ root elements	any number of	exactly one	optionally one	Twice		exactly one
51	Which of the following is an error in XML	nesting of elements	not declaring the version of XML	Xml files not ending with.xml extension	creatig more than one root element		creatig more than one root element
52	XML tags are _____	ignores case	case sensitive	reserved words	all uppercase only		case sensitive
53	XML document are _____ if it is syntactically correct	valid	invalid	well-formed	parsed		well-formed
54	_____ is required to parse the XML Document	XML-Parser	DTD file	Schema	XSL		XML-Parser
55	_____ parser is inbuild within IE5	sax	Xerces	JAPX	msxml		msxml
56	JAXP stands for	Java API For XML Parsing	Java and Xml Programming	Java Based XML Application Program	Parser		Java API For XML Parsing
57	SAX stands for	Stylesheet Application for XML	Simple API for XML	Schema API for XML	SGML And XML Programming		Simple API for XML
58	_____ consists of characters that may be represented in a document	ASCII	Markup text	Character Set	CDATA		Character Set
59	XML documents may contain _____characters	ASCII	Unicode	BCD	EBCDIC		Unicode
60	_____ consists fo characters of most worlds major languages.	ASCII	Unicode	BCD	EBCDIC		Unicode

UNIT-II
SYLLABUS

Document Type Definition – Parsers, Well-formed and valid XML documents – Element type declarations – Attribute declarations- Attributes Types. Schemas:– Schemas VS DTD's – MSXML Schemas, W3C XML Schema –Declaring Types and Elements –Attribute constraints and Defaults - Simple type - Empty elements -Mixed content elements – Creating Attribute Groups

CREATING MARKUP WITH XML

- XML is a subset of SGML(Standard Generalized Markup Language)
- XML basic syntax is similar to HTML's, the purpose of XML is different.
- XML is a technology for creating markup language to describe data of virtually any type in a structured manner. But HTML which limits the document author to a fixed set of tags.

INTRODUCTION TO XML MARKUP

- XML document are commonly stored in text files that end in the extension .xml.
- Any text editor can be used to create an XML document. Many software packages also allow data to be saved as XML documents.
- Simple example of xml document containing a message

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

<!--simple introduction to xml markup→

<mymessage>

<message>Welcome to XML</message>

</mymessage>
```

Line 1: Declared the version of xml ie 1.0.

Line 2: comment statements

Line 3, 5: Root element

Line 4: child element.

Note:

- Xml document contain exactly one root element the root element are the prolong of the XML document

- comment statements contain the same comment syntax as HTML.

PARSERS AND WELL-FORMED XML DOCUMENTS

- XML parser or XML processor is a software program which processes the XML document.
- The XML parser read the XML document, checks the syntax, reports any error and allows programmatic access to the document content.
- XML syntax required a single root element, a start tag and end tag for each element, properly-nested tags and attribute values in quotes.
- XML is case sensitive, so proper capitalization should be done.
- Parser can support the Document Object Model (DOM) and/or simple API for XML (SAX).
- SAX is used for accessing document's content programmatically using languages such as Java, Python and C etc
- A DOM-based parser builds a tree structure containing the XML document's data in memory.
- SAX-based parser processes the document and generates events. These events return data from XML document.

NOTE: Most of the XML parser is downloaded at no charge.

Eg Microsoft : msxml.

Apache XML Project's parser : Xerces.

Java API for XML parsing : JAXP.

IBM's parser XML for java : XMLAJ.

PARSING AN XML DOCUMENT WITH MSXML

A XML document contain data not formatting information. When an XML document is loaded into IE5,the document is parsed by msxml.

- The (-)sign and (+) sign are part of XML document.

- IE5 places either plus or minus sign next to all the element that contain one or more child elements.Because these element store other element known as container elements.
- A minus sign indicates that all child elements are visible. When a minus sign is clicked it become plus sign which collapse the container element and hides all its child element .

CHARACTERS

A character set consist of the characters that may be represented in a document.

- Character Set
  Character set consist of : carriage returns, line feeds and Unicode characters.

- Character VS Markup
  Markup text encloses on the angle brackets (< and>).Character data is the text between the start tag and end tag.

- White Space, Entity References and Built-in Entities
  -Spaces ,tabs,line feeds and carriage returns are character commonly called whitespace characters.

  -Depending on the application, insignificant whitespace characters may be collapsed into a single whitespace character o even remove entirely. This process is called normalization.

  -Almost all character can be used in an XML document, but the character ampersand(&),left angle bracket(<),right-angle bracket(>),apostrophe(‘) and double quote(“) are reserved in XML and may not be used in character data.

  -Entity reference is used to include these character for ampersand(&) begin with ampersand and ends with a semicolon(;).

  -XML provides build-in entities for ampersand(&),left-angle bracket(<),right-angle bracket(>),apostrophe(')and quotation mark(*quot;)

  -For ex to mark up the characters”<>&” in elemenet message we would write

  <message><>&</message>

Using Unicode in a XML Document

We can use Unicode in xml language for example below is used to print Arabic words in xml language.

```
<?xml version=”1.0”?>
<!- -Demonstrating Unicode - - >
<!DOCTYPE welcome SYSTEM “lang.dtd”>
<welcome>
<from>
<!- -Deitel and Associates- ->
&#1583; &#1575; &#1610; &#1578; &#1614; &#1604;
&#1571; &#1606; &#1583;
<!- -entity - ->
&assoc;
</from>
<subject>
<! - - Welcome to the world of Unicode - ->
        &#1571; &#1607; &#1604; &#1575; &#1611;
&#1576; &#1603; &#1605;
&#1601; &#1610; &#1616;
&#1593; &#1575; &#1604; &#1605;
<!- -entity- ->
&text;
</subject>
</welcome>
```

In this eg the third line gives the DOCTYPE , the name of the root element (welcome),the SYSTEM flag - which indicates that the DTD is located in an external file rather than the document itself and the name of the file (lang.dtd) containing the DTD.

MARKUP

XML markup consist of a start tag, content and an end tag

Eg <img src=”img.gif”></img>

This type of elements are called empty elements because it does not contain character.We can also write it as

- forward slash (/) is for termination.
- Attributes can be placed with the start tag.Attributes value are enclosed in code either single or double

<car doors=”4”/>

contains attribute whose value is 4.

Eg of markup

```
<?xml version=”1.0”?>
<!--usage of elements and attribute- ->
<?xml:stylesheet type=”text/xsl”href=”usage.xsl”?>
<book isbn=”999-9999-9-x”>
        <title>Deitel&amp;s XML Primer</title>
        <author>
                <firstName>paul</firstName>
                <lastName>Deitel</lastName>
        </author>
```

```
<chapters>
    <preface num="1" pages="2">Welcome</preface>
    <Chapter num="1" pages="4">Easy XML</chapter>
    <Chapter num="2" pages="2"> XML Elements ?
  </chapter>
    <appendix num="1" pages="9">Entities</appendix>
 </chapters>
<media type="CD"/>
</book>
```

In the above program author contain two child elements: firstName and lastName.Element chapter contain four child elements: preface, two chapter elements and appendix. Each of these elements has two attributes: num and pages.

CDATA SECTIONS

CDATA sections may contain text,reserved characters and whitespace characters.Character data in a CDATA section is not processed by the XML parser.A common use of a CDATA section is for scripting a code which often include the charcters &,<,>' and ".

Eg

```
<?xml version="1.0"?>
<!- -CDATA section containing c++ code- ->
<book title="c++ How to Program" edition="3">
  <sample>
    //c++ comment
    if(this-&gt;getx( ) &lt; 5 &amp;&amp; value[0]!=3)
      cerr &lt;&lt; this-&gt,displayError( );
```

```
</sample>
<sample>
        <![CDATA[
                //c++ comment
                if(this-->getX( ) <5 && value[ 0 ]!=3)
                        cerr<<this-->displayError( );
         ] ]>
</sample>
c++ how to program by Deitel &amp;amp; Deitel
</book>
```

Note: CDATA cannot contain the text]]> because it is used to terminate CDATA section.

XML NAMESPACES

XML allow document author to create their own tags, naming collisions can occur for example name of the subject in school and there may be subject in medicine.

Eg

<subject>Math</subject>

and

<subject>Thrombosis</subject>

Now the above two subjects can be differentiated by using namespace.

<school:Subject>Math<school:Subject>

and

<medical:Subject>Thrombosis</medical:subject>

*Both school and medical are namespace prefixes

*Each namespace prefix is tied to a uniform resource identifier(URL) that uniquely identifies the namespace

*Author can create there own namespace

eg

```
<?xml version=”1.0”?>
<!- -Namespace- ->
<directory xmlns:text=”urn:deitel:textInfo”
        xmlns:image=”urn:deitel:imageInfo”>
 <text:file filename=”book.xml”>
    <text:description>A book list</text:description>
<text:file>
<image:file filename=”funny.jpg”>
      <image:description>Afunny picture</image:description>
    <image:size width=”200” height=”100”/>
      </image:file>
</directory>
```

In order to place a namespace prefix in each element authors may specify a default namespace for an element and all of its child element.

eg

```
<?xml version=”1.0”?>
<!- -Namespace- ->
```

```
<directory xmlns:text="urn:deitel:textInfo"
        xmlns:image="urn:deitel:imageInfo">
  <file filename="book.xml">
      <description>A book list</description>
<file>
<image:file filename="funny.jpg">
      <image:description>Afunny picture</image:description>
      <image:size width="200" height="100"/>
      </image:file>
```

comparing the above program with the previous one the namespace prefix text is not present here.and the default namespace applies to all the elements contained in the directory

DOCUMENT TYPE DEFINITION (DTD)

Document type definitions define the XML document's structure. Provide the details of the elements, attributes which are permitted in the documents.

PARSERS, WELL-FORMED AND VALID XML DOCUMENTS

- Parser are generally classified as validating or nonvalidating.
- Validating: A validating parser is able to read the DTD and determine whether or not the XML document conforms to it. If conforms referred as valid if not the document is not well formed.
- Nonvalidating: A nonvalidating parser is able to read the DTD,but cannot check the document against the DTDfor conformity.If the document is syntactically correct,it is well formed.

DOCUMENT TYPE DECLARATION

- DTD is introduced into XML documents using the document type declaration (i.e. DOCTYPE).
- A document type declaration is placed in the XML document's prolog and begins with<!DOCTYPE and ends with>

- Document type declaration can be done in two way inside or outside the document.
- inside the document called internal subset

```
<!DOCTYPE myMessage[
   <!ELEMENT myMessage(#PCDATA)>
]>
```

myMessage is the name of the document type declaration. Anything inside the square brackets ([]) constitutes the internal subset.

Outside the document is called External subset. External subsets physically exist in a different file that typically ends with the .dtd extension, although this file extension is not required. External subset is specified using the keyword SYSTEM or PUBLIC.

Eg

```
<!DOCTYPE myMessage SYSTEM "myDtd.dtd">
```

PUBLIC keyword indicates that the DTD is widely used.

ELEMENT TYPE DECLARATIONS

- Elements are primary building block used in XML documents and are declared in a DTD with element type declarations(ELEMENT).
- <!ELEMENT myElement (#PCDATA)>
- The element name that follows ELEMENT is often called as Generic identifier.
- Keyword PCDATA specifies that the element must contain parsable character data.

Eg

```
<?xml version="1.0"?>
<!- -intro.xml- ->
<!- - Using an External subset- ->
<!DOCTYPE myMessage SYSTEM "intro.dtd">
```

<myMessage>

<message>Welcome to XML!</message>

</myMessage>

<!- - intro.dtd- ->

<!- -External declaration - ->

<!ELEMENT myMessage(message)>

<!ELEMENT message(#PCDATA)>

Sequences, Pipe Characters and Occurrence Indicators

- DTD allow the document author to define the order and frequency of child elements.
- Sequence: Comma(,)-called the sequence-specifies the elements must occur
  Eg

  <!ELEMET classroom(teacher,student)>

  Classroom must contain exactly one teacher element followed by exactly one student.

- Pipe characters: pipe (|) this is use to select from the elements one must be present.
  <!ELEMENT dessert(iceCream|pastry)

  This specifies the element dessert must contain either one icream element or pastry element

- Occurrence: An element frequency that is number of occurrence is specified by plus sign(+),asterisk(*)or question mark(?)

Occurrence indicator	**Description**
Plus sign(+)	An element can appear any number of times, but must appear at least once.(one or more times)
Asterisk (*)	An element is optional and if used, the element can appear any number of times (Zero or more times).

Question mark(?)	An element is optional and if used, the element can appear only once (zero or one time)

Eg 1) <!ELEMENT album(song+)>

Specifies that element album contains one or more song element.

Mark up will be given as

```
<album>
    <song>AR rehman beets</song>
    <song>Pray for me brother</song>
<album>
```

2) <!ELEMENT library(book*)>

Indicates library contain any number of elements, including the possibility of none at all.

Mark up will be given as

```
<library>
    <book>c++ how to program</book>
    <book>java the complete reference</book>
</libraray>
```

or

```
<library></library>
```

3) <!ELEMENT seat(person?)>

Indicates the element seat contains at most one person element

Mark up will be given as

<seat>

<person>Jackson</person>

<seat>

Or <seat></seat>

4) <!ELEMENT donutBox(jelly?,lemon*,((crème|sugar)+|glazed))>

specifies that element donutbox can have zero or one jelly elements, followed by zero or more lemon elements, followed by one or more crème or sugar elements or exactly one glazed element.

Mark up will be given as

<donutbox>

<jelly>grape</jelly>

<lemon>half-sour</lemon>

<lemon>sour</lemon>

<glazed>chocolate</glazed>

</donutbox>

and

<donutbox>

<sugar>semi-sweet</sugar>

<crème>whipped</crème>

<sugar>sweet</sugar>

</donutbox>

EMPTY,Mixed Content and ANY

Element must be further refined by specifying the content they contain.

EMPTY: keyword EMPTY declare empty elements. Empty element do not contain character data or child elements.

Eg <!ELEMENT oven EMPTY>

declares element oven to be an empty element.The markup for an oven element would appear as

<oven/>

Mixed Content: An element can also be declared as having mixed content.Such elements may contain any combination of elements and PCDATA. For example

<!ELEMENT myMessage (#PCDATA | message)*>

indicates the element myMessage contains mixed content.Markup will be given as

<myMessage>Here is some text,some

<message>other text</message> and

<message>even more text</message>

</myMessage>

ANY: An element declared as type ANY can contain any content,including PCDATA,elements or a combination of elements and PCDATA.Elements with ANY content can also be empty elements.

ATTRIBUTE DECLARATIONS

Attribute declaration specifies an attribute list for an element by using the ATTLIST attribute list declaration. An element can have any number of attributes.

<!ELEMENT x EMPTY>

<!ATTLIST x y CDATA #REQUIRED>

declare EMPTY element x. The attribute declaration specifies that y is an attribute of x keyword CDATA indicates that y can contain any character text except for the<,>,&,'and " characters

Eg

```
<?xml version = "1.0"?>
<!- - Declaring Attributes-->
<!DOCTYPE myMessage[
    <!ELEMENT myMessage ( message )>
    <!ELEMENT message(#PCDATA)>
    <!ATTLIST message id CDATA #REQUIRED>
]>
    <myMessage>
        <message id="445">
            Welcome to XML!
        </message>
    </myMessage>
```

Attribute Defaults(#REQUIRED,#IMPLIED,#FIXED)

Keyword #IMPLIED specifies that if the attribute does not appear in the element, then the application using the XML document can use whatever value(if any)it chooses.

keyword #REQUIRED indicates that the attribute must appear in the element.

Eg : markup is given as

<message>number<message>

When checked the DTD is declared as

<!ATTLIST message number CDATA #REQUIRED>

keyword #FIXED specifies that the attribute value is constant and cannot be different in the XML document.

<!ATTLIST address zip #FIXED "03115">

"03115" is the only value attribute zip can have.

ATTRIBUTE TYPE

- Attribute types are classified as either strings(CDATA),tokenized or enumerated.
- Tokenized attributes impose constraints on attribute values such as characters are permitted in an attribute name.
- Enumerated attribute is the most restrictive.They can take only one of the values listed in the attribute declaration.

Tokenized Attribute Type(ID,IDREF,ENTITY,NMTOKEN)

Tokenized attribute type ID uniquely identifies an element. Attributes with type IDREF point to elements with an ID attribute.A validating parser verifies that every ID attribute type referenced by IDREFis in the XML document.

eg

```
<?xml version =”1.0”?>
<!- - example for ID and IDREF values of attribute - ->
<!DOCTYPE bookstore[
        <!ELEMENT bookstore (shipping+,book+)
        <!ELEMENT shipping (duration)>
        <!ELEMENT Shippind shipID ID #REQUIRED>
        <!ELEMENT book (#PCDATA)>
        <!ELEMENT book shippedBy IDREF #IMPLIED>
        <!ELEMENT duration (#PCDATA)>
]>

<bookstore>
        <shipping shipID=”s1”>
```

<duration>2 to 4 days</duration>

</shipping>

<shipping shipID=”s2”>

<duration>1 day</duration>

</shipping>

<book shippedBy = “s2”>

Java how to program 3rd edition

</book>

<book shippedBy = “s2”>

C how to program 3rd edition

</book>

<book shippedBy = “s1”>

C++ how to program 3rd edition

</book>

</bookstore>

Attribute shipID as an ID type attribute book elements with attribute shippedBy of type IDREF.Attribute shippedBy points to one of the shipping elements by matching its shipID attribute.

ENTITY: A parser replaces the entity references with their values. For example

<!ENTITY digits “0123456789”>

For digits this entity might be used as follows

<useAnEntity>&digits;</useAnEntity>

The entity reference &digits; is replaced by its value resulting in

<useAnEntity>0123456789</useAnEntity>

These entities are called general entities.primary constrain placed on ENTITY attribute type is that they must refer to external unparsed entities.

Attribute type ENTITIES may also be used in a DTD to indicate that an attribute has multiple entities for its value. Each entity is separated by a space. For example

<!ATTLIST directory file ENTITIES #REQUIRED>

Markup will be given as

<directory file=”animations graph1 graph2”>

NMTOKEN: *name token value consist of letter,digits ,periods , underscore ,hyphens and colon characters. For example

<!ATTLIST sportsClub phone NMTOKEN #REQUIRED>

Which indicates the sportsclub contains a required NMTOKEN phone attribute.An markup example is given below

<sportsClub phone=”555-111-2222”>

Enumerated Attribute Types

- Declare a list of possible values an attribute can have.
- Enumerated values are separated by pipe characters.

For example

<!ATTLIST person gender (M|F) ”F”>

Declaration allows the value of M or F.A default value of F is also specified.

Example

<!ATTLIST person gender(M|F) #IMPLIED>

According to the name the gender will be decided if no gender is provided.

NOTATION is also an enumerated attribute type.for example

<!ATTLIST book reference NOTATION(JAVA|C)”C”>

The NOTATION of c can be declared as

<!NOTATION C SYSTEM

http://www.deitel.com/books/200/chtp3/chtp3_toc.htm>

Conditional Section

- DTDs provide ability to include or exclude declaration using conditional sections.
- Keyword INCLUDE specifies that declaration included, while Keyword IGNORE specifies that declaration are excluded.
  <![INCLUDE[

  <!ELEMENT name(#PCDATA)>

  ]]>

Directs the parser to include the declaration of element name

Similar conditional section

<![IGNORE[

<!ELEMENT message(#PCDATA)>

]]>

Directs the parser to ignore the declaration of element message.

- Conditional section are often used with entities
  <!ENTITY % reject "IGNORE">

  <!ENTITY % accept "INCLUDE">

  Declares entities reject and accept with the values IGNORE and INCLUDE.

- The entities is preceded by a percent (%) character they can be used Only inside the DTD in which they are declared. These types of

  Entities called parameter entities.

Whitespace Character

Whitespace is either preserved or normalized, depending on the context in which it is used.

Example

```
<! - - Documentation Whitespace parsing - ->

    <!ELEMENT whitespace( hasCDATA, hasID, hasNMTOKEN, hasEnumeration,
    hasMixed)>

    <!ELEMENT hasCDATA EMPTY>

    <!ATTLIST hasCDATA cdata CDATA #REQUIRED>

    <!ELEMENT hasID EMPTY>

    <!ATTLIST hasID id ID #REQUIRED>

    <!ELEMENT hasNTOKEN EMPTY>

    <!ATTLIST hasNMTOKEN nmtoken NMTOKEN #REQUIRED>

    <!ELEMENT hasEnumeration EMPTY>

    <!ATTLIST hasEnumeration enumeration (true | false ) #REQUIRED>

    <!ELEMENT hasMixed (#PCDATA | hasCDATA)*>

    ]>

    <Whitespace

        <hasCDATA cdata="    simple  cdata     "/>


        <hasID id = "      120 "/>


        <hasNMTOKEN nmtoken = "              hello "/>

        <hasEnumeration enumeration = "       true"/>


        <hasMixed>

            This is text.

            <hasCDATA cdata = "     simple     cdata"/>
```

This is some additional text.

</hasMixed>

</Whitespace>

The above program shows how white space is declared in each element. Major attribute in which white space used are ID,NMTOKEN , Enumeration ,Mixed.

SCHEMAS

*Alternative to DTDs called schemas for validating XML document.

*like DTDs schemas must be used with validating parsers.

*W3C XML schemas are in developing stage the well developed schemas are Microsoft XML schema.

SCHEMAS VS DTDS

DTD describes an XML document's structure—not its element content.

For example,

<quantity>5</quantity>

Contains character data. Element quantity can be validated to confirm that if does indeed contain content but its content cannot be validated to confirm what it is numeric ; DTDs do not provide such facilities.

< quantity>Hello</quantity>

Is also valid. With XML schema element quantity data can indeed be described as numeric. When the presiding markup example are validated against XML Schemas that specifies element quantities data must be numeric 5 confirm and hello fails.

*An XML document that confirms to a schemas document is schema valid and a document that does not conform is invalid.

*Unlike DTDs schema do not use the Extended Backus-Naur Form(EBNF)grammer instead schema use XML syntax.

Microsoft XML Schema:Describing Elements

*Elements are primary building blocks used to create XML document. In Microsoft XML schemas, element ElementType defines an element.

*ElementType contains attributes that describe the element's content, data type, name, etc.

Example

```
<?xml version =  "1.0"?>
<!- -Microsoft XML schema showing the ElementType- -!>
<schemas xmlns = "urn:schemas-microsoft-com:xml-data">
        <ElementType name = "message" content = "textonly" model = "closed">
        <description>Text message</description>
        </ElementType>
        <ElementType name = "greeting" model = "closed" content
        ="mixed" order = "many">
        <element type = "message"/>
        </ElementType>
        <ElementType name = "myMessage" model = "closed" content
        ="eltOnly" order = "seq">
        <element type = "greeting" minoccurs  = "0" maxoccurs  = "1"/>
        <element type = "message" minoccurs  = "1" maxoccurs  = "*"/>
        </ElementType>
        </schema>
```

The xml document given as

```
<?xml version = "1.0"?>
<! - - Introduction to Microsoft XML Schema - - >
        <myMessage xmlns = "X – schema:intro-schema.xml">
        <greeting>Welcome to XML schema!
```

<message>This is the first message</message>

</greeting>

<message>This is the second message</message>

</myMessage>

ElementType element attributes.

Attribute Name	::Description
Content	Describes the elements content the valid values for this attribute are empty,eltonly,textonly and mixed.
Dt:type	Defines the elements data type.Data types exist foe real number,integers,Boolean,etc.
Name	The elements name.this is a required attribute.
Model	Specifies whether elements not defines in the schema are permitted in the element.valid values open(outside the schema) and closed(inside the schema).
Order	Specifies the order in which the child elements must occur.the valid values for the attribute are one(only one child element),seq(child elements can appear in any order in which they are defined) and many(child element can appear in any order)

Element ElementType's child elements.

Element Name	::	Description
Description		provides a description of the ElementType
Datatype		Specifies the data type for the Elementype element.
Element		specifies a child element by name.
Group		groups related element elements are defines their order and frequency.
AttributeType		defines an attribute.

Attribute	Specifies an AttributeType for an element.

Element element attributes

Attribute Name	:: Description
Type	A required attribute that specifies achild elements name
Minoccurs	specifies the minimum number of occurrences an element can have.The valid value are 0 and 1.the default value is 1.
Maxoccurs	specifies the maximum number of occurrences an element can have.The valid values are 1 and *.the default value is 1 unless the ElementTypes content attribute is mixed.

Element group's attributes

Attribute Name	:: Description
Order	specifies the order in which the elements occur. The valid values are one ,seq and many.
Minoccurs	specifies the minimum number of occurrences an element can have. the values are 0 and 1.
Maxoccurs	specifies the maximum number of occurrences an element can have.the valid value is 1 and *(the element can occur any number of times).

Microsoft XML Schema:Describing Attributes

ElementAttributeTypes attributes

Attribute Name	:: Description
Default	specifies the attributes default value.
Dt:type	defines the elements data type.
Dt:values	contains an enumeration data types values.
Name	the attribute name.this is a required attribute.
Required	indicates whether the attribute is required.the valid value are yes and no.

Element attributes attributes

Attribute Name	:: Description
Default	specifies the attributes default value.this value overrides the value defined in the AttributeType element.
Type	specifies the name of the AttributeType for the attribute.this is a required attribute.
Required	indicates whether the attribute is required.valid values for this attribute are yes and no.the default value is no.

Microsoft XML schema:Data Types

Datatype	Description
Boolean	0(false) or 1(true).
Char	A single character(eg “D”)
String	A series of characters(eg “deitel”)
Float	A real number(eg 123.4656)
Int	A whole number(eg 5)
Date	A date formatted as YYYY-MM-DD(eg 2000-04-25)
Time	A time formatted as HH-MM-SS(eg 14:30:00)
Id	text that uniquely identifies an element or attribute.
Idref	A reference to an id.
Enumeration	A series of values from which only one may be chosen.

W3C XML SCHEMA

The W3C is developing an XML Schema specification, which is a Candidate Recommendation. Latest specification of W3C XML schemas was given in www.w3.org/XML/Schema.

Microsoft XML schema document commonly use the .xml extension and W3C XML schema commonly use the .xsd extension.

W3C XML schemas elements defines elements. Attributes name and type specify the elements name and data type, respectively. Attribute ref references the existing element definition for message. This indicates that greeting can have element message as a child element.

W3c XML Schema document

```
<?xml version =”1.0”?>
<!- - example of E3C XML schema - ->
<xsd:schema xmlns:xsd = http://www.w3.org/2000/10/XMLSchema>
    <xsd:element  name = “message” type = “xsd:string”/>
    <xsd:element  name = “greeting” type = “greetingType”/>
    <xsd:complexType  name = “greetingType” content = “mixed”>
        <xsd:element ref  =  “message”/>
    </xsd:complexType>
    <xsd:element name =  “myMessage” type = “myMessageType”/>
        <xsd:complexType name = “myMessageType”>
        <xsd:element ref = ”greeting”  minoccurs = “0” maxoccurs  = “1”/>
        <xsd:element ref = ”message”  minoccurs = “0” maxoccurs  = “unbounded”/>
        </xsd:complexType>
</xsd:schema>
```

Documents that confirm xml-schema.xsd.

```
<?xml version  = "1.0"?>
<- -Introduction to W3C XML schema- ->
<myMessage>
        Xmlns:xsd = http://www.w3.org/2000/10/XMLSchema-instance
        Xsd:noNamespaceSchemaLocation = "xml-schema.xsd">
        <greeting>Welcome to W3C XML schema!</greeting>
        <message>This is a message</message>
        <message>This is another message</message>
        </myMessage>
```

POSSIBLE QUESTIONS

PART B QUESTIONS
(EACH QUESTION CARRIES SIX MARKS)

1. Explain how to declare elements in MSXML Schema.
2. Explain how to declare attributes in MSXML Schema.
3. Explain the 'Characters' used in XML Documents.
4. Explain about the Attribute description in DTD.
5. Explain how the Elements are described in DTD.
6. Explain the different Attribute Types present in MSXML schema.
7. Explain how to describe attributes in MSXML schema.
8. Explain the purpose of Occurrence Indicators in DTD.
9. Explain all the data types used in MSXML Schema with an example.
10. Illustrate with example how to describe the elements in W3C Schema.

PART C QUESTIONS
(EACH QUESTION CARRIES TEN MARKS)

1. Explain the process and with suitable examples i) XLink ii) XPointer
2. Explain about the DOM Traverse Node.
3. Write a program to storing XML Files on the server and explain in detail.
4. Write a java program to parsing an XML file using SAX.
5. Write a program to view an XML Student details and display the Student Details with an XSLT Stylesheet.

Karpagam Academy of Higher Education

Department of Computer Applications

Part-A Questions

Subject Code : 17CAP404W **Subject : XML** **Class : II MCA**

UNIT - II							
S.No.	**Questions**	**Choice 1**	**Choice 2**	**Choice 3**	**Choice 4**		**Answer**
1	Space, tabs, line feeds and carriage returns are commonly called	CDATA	PCDATA	White Space characters	Build-in Entities		White Space characters
2	Insignificant Whitespace characters are collapsed into a single whitespace character. This is called ___	trimming	collapsing	parsing	normalising		normalising
3	< > " these are called	Markup text	CDATA Section	Built -in entities	White Space Characrers		Built -in entities
4	All Element names are enclosed within _____	< >	( )	{ }	[ ]		< >
5	Each _____contains a starting tag and an ending tag	Element	Attribute	Entity	DATA		Element
6	If an element tag starts with < and ends with /> it is _____	root element	parent element	child element	empty element		empty element
7	Processing Instructions are delimited by	< >	<!-- -->	<? ?>	< />		<? ?>
8	___ is a name value pair	Element	Attribute	Entity	Data		Attribute
9	The values of an attribute is enclosed within	< >	" "	/* */	( )		" "
10	A comment in XML lies within _____	< >	" "	/* */	<!-- -->		<!-- -->
11	_____ may contain text, reserved characters and even multiple whitespace characters	Markup text	CDATA Section	Built -in entities	Text contained in Element		CDATA Section
12	Naming Collisions can be resolved by using _____	Markup text	CDATA Section	Built -in entities	Namespaces		Namespaces
13	The Keyword _____ is used to create namespace prefixes	XML	msxml	xmlns	xns		xmlns
14	Each namespace prefix is tied toa a _____	URI	DTD	XML	DOM		URI
15	Child elements of a default namespace	are prefixed with xml	are prefixed with msxml	are prefixed with xmlns	do not need a prefix		do not need a prefix
16	The set of document type declarations inside an xml document is called the_____	public file	external subset	internal subset	system file		internal subset
17	_____ are the basic building blocks of an xml document	Elements	Attributes	Tags	Text		Elements
18	Elements are declared with the _____ type declaration in the DTD.	ELEMENT	PCDATA	ATTLIST	ENTITY		ELEMENT
19	Keyword _____ indicates that an element contains parsable character data.	CDATA	PCDATA	ATTLIST	ENTITY		PCDATA

20	In an element type decalaration, the pipe character (|) indicates that the element can contain ____ of the elements indicated.	all	any one	each	many		any one
21	Attributes are declared using the _____ type	ELEMENT	PCDATA	ATTLIST	ENTITY		ATTLIST
22	Keywod _____ indicates that the attribute can only take a specific value that has been defined in the DTD.	# IDREF	# REQUIRED	# FIXED	# IMPLIED		# FIXED
23	ID, IDREF, ENTITY,NMTOKEN are all types of _____	attribute defaults	attribute declarations	enumerated attributes	tokenized attributes		tokenized attributes
24	# REQUIRED, # IMPLIED, # FIXED are all	attribute defaults	attribute declarations	enumerated attributes	tokenized attributes		attribute defaults
25	Which of these is not an attribute default	# IDREF	# REQUIRED	# FIXED	# IMPLIED		# IDREF
26	Which of these is not an attribute type	CDATA	Enumerated	Tokenized	PCDATA		PCDATA
27	Which of these is a tokenized attribute	ID	NMTOKENS	ENTITIES	ID,NMTOKENS, ENTITIES		ID,NMTOKENS, ENTITIES
28	The % character is used to declare _____	Elements	parameter entity	attribute defaults	enumneration		parameter entity
29	Conditional sections of DTDs are often used with ___	elements	entities	attributes	parameters		entities
30	An XML document is said to be _____ if it does not reference an external DTD.	valid	well-formed	standalone	parsable		standalone
31	DTDs can be introduced into XML documents by using _____	DOCTYPE	ATTRTYPE	ELEMENT	PCDATA		DOCTYPE
32	If the DTD is defined outside the XML document it is called _____	public file	external subset	internal subset	system file		external subset
33	External DTDs are referenced using the _____ keyword	EXTERNAL	DOCTYPE	SYSTEM	SUBSET		SYSTEM
34	_____ are defined using Extended Backus-Naur Form (EBNF) grammar	DTD	DOM	Schema	XML		DTD
35	_____ use XML syntax instead of EBNF grammar	DTD	DOM	Schema	XML		Schema
36	The Document Type Declaration is placed in the ___ of the XML	epilogue	prolog	boby	xml tag		prolog
37	The External subset exists in a file that commonly have the extention _____	.xml	.xsl	.dom	.dtd		.dtd
38	The _____of the DTD is visible only with in the document it resides	internal subset	xml-declarations	external subset	CDATA Sections		internal subset
39	The element name that follows ELEMENT keyword in DTD is called the _____	root element	generic identifier	element type indicator	Elements		generic identifier

40	The element's frequency is specified by using the _____	element type indicator	occurance indicator	generic identifier	PCDATA		occurance indicator
41	_____ specifies that an element must occur atleast once and may occur more that once also (one or more times)	+	*	?	-		+
42	_____ specifies that an element is optional, and if it occurs it may appear only once (0 ot 1 time)	+	*	?	-		?
43	_____ indicates that an element is optional, and if it occurs it may appear any number of times (0 or more)	+	*	?	-		*
44	<!ELEMENT myMessage (#PCDATA | Message)*> indicates that the element myMessage may contain_____	either parsable data or element Message	mixed content	Multiple occurance of Message element only	Variable message		Multiple occurance of Message element only
45	Schema documents use _____ syntax	EBNF	XML	Java	DTD		XML
46	The collection of DTD's and Schemas for a variety of applications available on the Web are called _____	recommendations	repositories	packages	Application Interfaces		repositories
47	_____ are expected to replace DTD's for describing XML structure.	CSS	XSL	DOM	Schemas		Schemas
48	_____ is the root element of every MS-XML Schema documents	<Xml>	<Root>	<Schema>	ANY		<Schema>
49	In MS-XML Schema _____ defines an element	element	ElementType	ELEMENT	xml:Element		ElementType
50	In MS-XML Schema_____ is used to refer an element defined by ElementType	element	ElementType	ELEMENT	xml:Element		element
51	Element 'Schema' contains 3 elements, they are ____	ElementType,AttributeType and Comments	ElementType,ElementName and Description	ElementType,AttributeType and EntityType	ElementType,AttributeType and description		ElementType,AttributeType and description
52	AttributeType defines an _____	attribute defaults	attribute	attribute data type	type of the attribute		attribute
53	_____ is used to reference fragments of a document	Xlink	Xpath	Xpointer	Xbase		Xpointer
54	_____ provide a method for including XML documents within other XML documents	Xinclude	Xpath	Xpointer	Xbase		Xinclude
55	Links between remote resources and local resourses are known as _____ links	simple	extended	inbound	inline		inbound
56	Xlink attribute _____ contains a human readable description of a link	name	title	description	label		title

57	An Xpath node is called a _____ in Xpointer	element	location	link	node		location
58	_____ allows the document author to change the base URI for any relative links in a document	Xinclude	Xpath	Xpointer	Xbase		Xbase
59	Xlink attribute _____ describes the relationship between resourses	arcrole	relate	locator	link		arcrole
60	CDF stands for	Common Document Format	Channel Delivery Format	Common Delivery Format	Channel Definition Format		Channel Definition Format

UNIT-III
SYLLABUS

Document Object Model: DOM implementations – DOM with JavaScript – Components- Creating nodes – Traversing the DOM. Simple API for XML: DOM vs SAX – SAX based Parsers.

DOCUMENT OBJECT MODEL (DOM)

INTRODUCTION

XML documents, when parsed, are represented as a hierarchical tree structure in memory. This tree structure contains the document's elements, attributes, content, etc. XML was designed to be a live, dynamic technology- a programmer can modify the contents of the tree structure, which essentially allows the programmer to add data, remove data, query for data, etc. in a manner similar to a database.

The W3C provides a standard recommendation for building a tree structure in memory for XML documents called the XML Document Object Model(DOM). Any parser that adheres to this recommendation is called a DOM-based parser. Each element, attribute, CDATA section, etc., in an XML document is represented by a node in the DOM tree. For example, the simple XML document.

```
<?xml version="1.0">
<message from ="paul" to ="tem">
<body>hi,tem!</body>
</message>
```

Results in DOM tree with several nodes. One node is created for the message element. This node has a child node that corresponds to the body element. The body element also has a child node that corresponds to the text Hi, Tem!. The form and to attributes of the message element also have corresponding nodes in the DOM tree.

A DOM-based parser exposes (i.e., makes available) a programmatic library-called the DOM Application Programming Interface (API)-that allows data in a XML document to be accessed and modified by manipulating the nodes in a DOM tree.

Another API-JDOM-provides a higher-level API than the W3C DOM for working with XML documents in java. Because JDOM is an API that is specific to the java programming language, it can take advantage of features in java that make it easier to program, JDOM is still in the early stages of development.

In order to use t he DOM API, programming experience is required. Although the DOM API is available in many languages (e.g., C, java, VBScript, etc.).

DOM IMPLEMENTATIONS

DOM-based parsers are written in a variety of programming languages and are usually available for download to at no charge. Many applications (such as Internet Explorer 5) have built-in parsers.

DOM WITH JAVA SCRIPT

To introduce document manipulation with the XML Document Object Model, we begin with a simple scripting example that uses JavaScript and Microsoft's msxml parser. This example takes an XML document that marks up an article and uses the DOM API to display the document's elements names and values. The below java script code that manipulates this XML document and displays its content in an HTML page.

<script type = "text/javascript" language = "JavaScript">

Parser	Description
JAXP	Sun Microsystem's java API for XML parsing(JAXP is available at no charge form java.sun.com/xml
XML4J	IBM's XML parser for java (XML4J) is available at no charge from www.alphaworks.ibm.com/tech/xml4j
Xerces	Apache's Xerces java parser is available at no charge from xml.apache.org/xerces
Msxml	Microsoft's XML parser(msxml)version 2.0 is built-into Internet Explorer5.5,version 3.0 is also available at no charge from msdn.microsoft.com/xml.
4DOM	4DOM is a parser for the python programming language and is available at no charge from fourthought.com/4suite/4DOM
XML::DOM	XML::DOM is a perl module that we use to manipulate xml documents using perl. For additional information. Visit www-4.ibm.com/software/developer/library/xml-perl2.

Example:

```
<?xml version = "1.0" ?>
<!—Article formatted with XML-->
<article>
<title>simple XML</title>
<data>December 6,2000</data>
<author>
  <fname>Tem</fname>
  <lname>Nieto</lname>
</author>
<summary>XML is pretty easy</summary>
<content>Once you have mastered HTML,XML is easily learned. You
    Must remember that XML is not for displaying information but for
    managing information.
```

</content>
</article>

Is the opening script tag, which the document author to include scripting code. Attribute type indicates that the script element is of media type text/javascript. JavaScript is the most popular client-side(e.g. browser) scripting language used in industry. If the browser does not support JavaScript, script's contents are treated as text. Attribute language indicates to the browser that the script is written in the javascript scripting language.
Example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
    http://www.w3.ort/TR/html4/strict.dtd>
<html>
<!—DOM with JavaScript   -- >
<head>
   <title> A DOM Example</title>
</head>
<body>
<script type="text/javascript" language= "JavaScript">
Var xmldocument=new ActiveXObject("Microsoft.XMLDOM");
xmlDocument.load("article");
//get the root element
Var element = xmlDocument.documentElement;
document.writeln("<p>Here is the root node of the document:");
document.writeln("<strong>"+element.nodeName+"</strong>");
document.writeln("</p><ul>");
//traverse all child nodes of root element
For ( I =0; i< element.childNodes.length;I++)
{ var curNode=element.childNodes.item(i);
//print node name of each child element
document.writeln("<li><strong>"+curNode.nodeName+"</strong></li>");
}
//get the first child node of root element
Var curfrentNode = element.firstChild;
Document.writeln("<p>The first child of root node is:");
Document.writeln("<strong>"+currentNode.nodeName+"</strong>");
Document.writeln("<br>whose next sibling is:");
//get the next sibling of first child
Var nextsib= currentNode.nextSibling;
Document.writeln("<strong>" +nextsib.nodeName+"</strong>element is:");
Var value = nextsib.firstChild;
//print the text value of the sibling
Document.writeln("<em>"+value.nodeValue+"</em>");
Document.writeln("<br>parent node of ");
Document.writeln("string>"+nextsib.nodeName+"</strong>is: ");
```

```
Document.writeln("<strong>"+nextsib.parentNode.nodeName+"</strong></p>");
</script>
</body>
</html>
```

Var xmlDocument = new ActiveXObject("Microsoft.XMLDOM");

Instantiates a Microsoft XML Document Object Model and assigns it to reference xmlDocument. This object represents an XML document (in memory) and provides methods for manipulating its data. The statement simply creates the object, which does not yet refer to any specific XML document.

xmlDocument.load("article.xml");

calls method load to load article.xml into memory. This XML document is parsed by msxml and stored in memory as a tree structure.

Var element = xmlDocument.documentElement;

Assigns the root element (i.e. article) to variable element. Property document element corresponds to the document's root element. The root element is important because it is used as a reference point for retrieving child elements, text, etc.

Document.writeln(""+element.nodeName +"");

Place the name of the root element in a strong element and write it to the browser where it is rendered. Property nodename corresponds to the name of an attribute, element, etc. which are collectively called nodes. In this particular case, element refers to the root node named article.

For(i=0;i<element.childNodes.length;i++)

Uses a for loop to iterate through the root node's child nodes (accessed using property childNodes). Property length is used to get the number of child nodes of the document element.

Individual child nodes are accessed using the item method. Each node is given an integer index (starting at zero) based on the order in which they occur in the XML document.

Var curNode=element.childNodes.item(i);

Call method item to return the child node identified by the index i. this node is assigned to variable curNode.

Var currentNode = element.firstchild;

Retrives the root node's first child node (i.e.,title) using property firstchild. This expression is a more concise alternative to

Var currentNode=element.childNodes.item(0);

Nodes at the same level in adocument(i.e., that have the same parent node) are called siblings. For example, title, date ,author ,summary and content are all sibling nodes. Property nextsibling returns a node's next sibiling.

Var nextsib= currentNode.nextsibling;

Assings currentNode's (i.e., title from line 45) next sibling (i.e., date) to nextsib;

In addition to elements and attributes, text(e.g; simple xml in line 8)is also a node.

Var value=nextsib.firstchild;

Assigns nextsib's (i.e., date) first child node to value. In this case, the first child node is a textnode. On line 63 the nodevalue method retrives the value of this text node. The value of a text node's value is the text it contains. Element nodes have a value of null (i.e., the absence of a value).

Document.writeln(""+nextsib.parentNode.nodeName +"<p>");

Retrieve and display nextsib's (i.e., date) parent node (i.e., article). Property parentNode returns a node's parent node.

DOM COMPONENTS:

In this section, we will use java, JAXP and the XML-related java packages to manipulate an XML document. Before discussing our first java-based example, we summarize several important DOM classes, interfaces and methods. Due to the number of DOM objects and methods available, we provide only a partial list of these objects and methods.

Class/Interface	Description
Document interface	Represents the XML document's top-level node, which provides access to all the document's nodes-including the root element.
Node interface	Represents an XML document node.
NodeList interface	Represents a read-only list of node objects.
Element interface	Represents an element node. Derives from node
Attr interface	Represents an attribute node. Derives from node.
characterData interface	Represents character data. Derives from node.
Text interface	Represents a text node. Derives from characterData
Comment interface	Represents a comment node. Derives from characterdata
processingInstruction interface	Represents a processing instruction node. Derives from node.
CDATASection interface	Represents a CDATA section. Derives from text.

The document interface represents the top-level node of an XML document in memory and provides a means of creating nodes and retrieving nodes. The methods of class XmlDocument, including the methods inherited from Document. Class XmlDocument is part of the JAXP internal APIs and its methods are not part of the W3C DOM recommendation.
Interface Node represents an XML document node.

Method Name	Description
createElement	Creates an element node.
createAtrribute	Creates an attribute node.
createTextNode	Creates a text node.
createComment	Creates a comment node.
createProcessingInstruction	Creates a processing instruction node.
createCDATASection	Creates a CDATA section node.
getDocumentElement	Returns the document's root element.
appendChild	Appends a child node.
getChildNodes	Returns the child nodes.
createXmldocument	Parses an XML document.
appendChild	Appends a child node.
cloneNode	Duplicates the node.
getAttributes	Returns the node's attributes
getChildNodes	Returns the node's child nodes.
getNodeName	Returns the node's name.
getNodeType	Returns the node's type.
getNodeValue	Returns the node's value.
getparentNode	Returns the node's parent.
hasChildNodes	Returns true if the node has child nodes.
removeChild	Removes a child node from the node.
replaceChild	Replaces a child node with another node.
setNodeValue	Sets the node's value.
insertBefore	Appends a child node in front of a child node.

NodeType	Description
Node.ELEMENT_NODE	Represents an element node.
Node.ATTRIBUTE_NODE	Represents an attribute node.
Node.TEXT_NODE	Represents a text node.
Node.COMMENT_NODE	Represents a comment node.
Node.PROCESSING_INSTRUCTION_NODE	Represents a processing instruction node.
Node.CDATA_SECTION_NODE	Represents a CDATA section node.

Simple example to replace an existing text node:

```
Import  java.io.*;
Import org.w3c.dom.*;
```

```
Import javax.xml.parsers.*;
Import com.sum.xml.tree.XmlDocument;
Import org.xml.sax.*;
Public class ReplaceText{
Private Document document;
Public ReplaceText()
{
try{
//obtain the default parser
DocumentBuilderFactory factory=DocumentBuilderFactory.newInstance();
//set the parser to validating
factory.setValidating(true);
DocumentBuilder builder=factory.newDocumentBuilder();
//set error handler for validating errors
Builder.setErrorHandler( new MyErrorHandler() );
//obtain document object from XML document
Document=builder.parse( new File("intro.xml"));
//fetch the root node
Node root=document.getDocumentElement();
If(root.getNodeType()==Node.ELEMENT_NODE){
Element myMessageNode=(Element)root;
NodeList messageNodes=myMessageNode.getEmenetsByTagName("message");
If(messageNodes.getLength()!=0){
Node message = messageNodes.item(0);
//create a text node
Text newText =document.createTextNode("new changed message");
//get the old text node
Text oldText=(Text) message.getChildNodes().item(0);
//replace the text
message.replaceChild(newText,oldText);
}
}
((XmlDocument)document).write(new FileOutputStream("intro1.xml"));
}
Catch(SAXParseException spe){
System.err.printLn("Parse error: "+spe.getMessage());
System.exit(1);
}
Catch(SAXException se){
Se.printStackTrace();
}
Catch(FileNotfoundException fne){
System.err.println("file\'intro.xml\' not found");
System.exit(1);
}
```

```
Catch(Exceotuib e){
e.printStackTrace();
}
}
Public static void main(String args[])
{
ReplaceText d= new ReplaceText();
}
}
```

```
Import  java.io.*;
Import org.w3c.dom.*;
Import javax.xml.parsers.*;
Import com.sum.xml.tree.XmlDocument;
Import org.xml.sax.*;
```

Import(i.e., specify the location of) the classes needed by the program. The java programming language provides a feature called packages that groups a series of related series of related java files (e.g. compiled .class files). For example, the first import statement indicates that our program uses some classes from the package java.io. Sun microsystem, the creator of java, provides several packages related to XML. Package org.w3c.dom provides the DOM-API programmatic interface. Packages javax.xml.parsers provides classes related to parsing an XML document.

Package com.sun.xml.tree contains classes and interfaces from sum micro system's internal XML API, which provides features currently not available in the xml 1.0 recommendation. Documentation for sun's internal APIs can be found at

Java.sun.com/xml/docs/api

A DOM based parser may use an event based implementation to help create the tree structure in memory. A popular event based implementation is called the simple API for XML (SAX). The main SAX feature we discuss in this chapter are exceptions thrown by the default parser used by the JAXP. An exception occurs when an error is encountered in a program. The error may be caught by an exception handler and processed, ensuring the program does not terminate abnormally. Package **org.xml.sax** provides the SAX programmatic interface.

Private Document document;
Declares Document reference document. This reference is assigned an object that represents the document root.

JAXP uses the DocumentBuilderFactory class to create a DocumentBuilder object. Class DocumentBuilder provides a atandard interface to an XML parser. JAXP can be configured to use many different XML parsers, such as the Apache Group's Xerces and AIBM's XML4J. JAXP also has its own parser built in, which is used by default. The documentbuilderfactory produces an appropriate DocumentBuilder object for the currently configured XML parser.

DocumentBuilderFactory factory=DocumentBuilderFactory.newInstance();

Create and assign a DocumentBuilderFactory object to reference factory

Factory.setValidating(true);

Indicates that a validating parser is being used by passing the value true as an argument to method setValidating.

DocumentBuilder builder = factory.newDocumentBuilder();

Creates a new DocumentBuilder object and assigns it to reference builder. This object provides and interface for loading and parsing XML documents.
Builder.setErrorHandler(new MyErrorHandler());

Specifies that a MyErrorHandler object provides methods for handling exceptions related to parsing.

Document = builder.parse(new File("intro.xml"));
Calls method parse to load and parse the XML document stored in the file intro.xml. if parsing is successful, a Document object is returned that contains nodes representing each part of the intro.xml document. If parsing fails, a SAXException is thrown.
Node root = deocument.getDocumentElement();
Calls method getDocumentElement to get the document's root node and assign it to node reference root.

If (root.getNodeType() == Node.ELEMENT_NODE) {

Texts if the root element is an element node. Method getNodeType is called to retrieve the node's type.

Element myMessageNode = (Element) root;

Down casts root from a superclass Node type to an Element derived type. As mentioned earlier, class element inherits from class node. By down casting, this allows element mymessagenode to be assigned the object referenced by root. Methods specific to class element can now be called on the object using the mymessagenode reference.

NodeList messageNodes = myMessageNode.getElementByTagname("message");

Get a list of all the message elements in the xml document using method getElementsByTagName. Each element is stored as an item in a node list. The first item added is stored at index 0, the next at index 1, and so forth. This index is used to access an individual item in the nodelist.

If(messageNodes.getLength()!=0){

Determines if the NodeList contains at least one item by calling method getLength and testing the value returned from it against zero.

Node message = messageNode.item(0);

Assigns the first NodeList to Node reference message. Method item returns an individual node from the NodeList. In this case, the first Node is returned.

Text newText = document.createTextNode("New changed message");

Use the createTextNode method to create a text node that contains the text new changed message. This node exists in memory independent of the XML document. It has not been inserted into the document yet. Interface text represents an element or attribute's character data.

Text oldText = (Text) message.getChildNodes().item(0);

Get the first child node of the message element which is a text node containing the text welcome to xml. Because item item returns an object of superclass type object, a downcast to text is performed.

Message.replaceChild(newText,oldText);

Calls method replaceChild to replace the Node referenced by the second argument with the node referenced by the first argument. The XXXXML document has now been modified element message now contains the text New changed message.

CREATING NODES:

The majority of XML markup presented up to this point has been "hand coded" . using the DOM,XML documents can be created in an automated way through programming.

Building an xml document with the DOM

```
Import java.io.*;
Import org.w3c.dom.*;
Import org.xml.sax.*;
Import javax.xml.parsers.*;
Import com.sun.xml.tree.XmlDocument;
Public class BuildXml {
  Private Document document;
Public BuildXml()
  {
```

```
   DocumentBuilderFactory factory=DocumentBuilderFactory.newInstance();
Try{
//get DocumenBuilder
DocumentBuilder builder= factory.newDocumentBuilder();
//create root node
Document =builder.newDocument();
}
Catch(ParserConfigurationException pce){
Pce.printStackTrace();
}
Element root = document.createElement("root");
Document.appendchild(root);

//add a comment to xml document
Comment simpleComment = document.createComment("This is a simple contact list");
root.appendChild(SimpleComment);

//add a child element
Node contactNode=createcontactNode(document);
Root.appendChild(contactNode);

//add a processing instruction
ProcessingInstruction pi=document.createProcessingInstruction("myinstruction","action silent");
root.appendChild(pi);

//add a CDATA section
CDATASection cdata=document.createCDATASection("I can add<,>, and?");
Root.appendChild(cdata);
Try{
//write the xml document to a file
{(xmlDocument)document).write(new FileOutputStream("mydocument.xml"));
}
Catch(IOException ioe){
Ioe.printStackTrace();
}
}
Public Node createContactNode(Document document)
{
//create firstname and lastname elements
Element firstName = document.createElement("firstName");
Element lastname=document.createElement("LastName");

firstName.appendChild(document.createTextNode("sue"));
lastName.appendChild(document.createTextNode("green"));
```

```
//create contact element
Element contact = document.createElement("contact");

//create an attribute
Attr genderAttribute = document.createAttribute("gender");
genderAttribute.setValue("F");

//append attribute to contact element
Contact.setAttributeNode(genderAttribute);
Contact.appendChild(firstname);
Contact.appendChild(lastname);
Return contact;
}
Public static void main(String args[])
{
BuildXml buildXml = new BuildXml();
}
}
```

DocumentBuilderFactory factory= DocumentBuilderFactory.newInstance();

Create and assign a documentbuilderfactory object to reference factory. Class DocumentBuilderFactory is used used to obtain an instance of a parser in this particular case, the default JAXP parser.

DocumentBuilder builder=factory.newDocumentBuilder();

Create a new DocumentBuilder object and assign it to reference builder. This object provides facilities for loading and parsing XML documents.

Document = builder.newDocument();

Calls method newDocument to create a new Document object. We will use the document object returned by newDocument to build an XML document in memory.

Element root = document.createElement("root");
Document.appendChild(root);

Create an element named root and append it to be document root. Because this is the first element appended, it is the root element of the element.

Comment simpleComment = document. createComment("this is a simple contact list");
Root.appendChild(simpleComment);

Create a comment node using method createComment and append the node as a child of element root.

Node contactNode = createContactNode(document);
Calls programmer-defined method createContactNode to create the contact element.

ProcessingInstruction pi=document.createProcessingInstruction("myInstruction","action silent");

Create a processing instruction node. The first argument passed to createProcessingInstruction is the target myInstruction and the second argument passed is the value action silent.

CDATASection cdata=document.createCDATASection("I can add <,>, and ?");

Create a CDATA section, which is appended to element root.

public Node createContactNode(Document document)

define method createContactNode that returns a node object. This method creates a contact element node and returns it. The returned node is appended to the root element.

Element firstName=document.createElement("firstname");
Element lastName=document.createElement("lastname");

Create element firstname and lastname which have their text values.

Attr genderAttribute = document.createAttribute("gender");
genderAttribute.setValue("F");

create attribute gender using method createAttribute and assign it a value using Attr method setValue.

Contact.setAtrributeNode(genderAttribute);

Assigns the attribute to the contact elementnode using method setAttributeNode.

TRAVERSING THE DOM

In this section, we demonstrate how to use the DOM to traverse an XML document. We present a java application that outputs element nodes, attribute nodes and text nodes. This application takes the name of a XML document from the command line.

Example:

```
Import java.io.*;
Import org.w3c.dom.*;
Import org.xml.sax.*;
Import javax.xml.parsers.*;
Import com.sun.xml.tree.XmlDocument;
Public class TraverseDOM{
    Private Document document;
Public TraverseDOM(String file)
{
try{
//obtain the default parser
DocumentBuilderFactory factory=DocumentBuilderFactory.newInstance();
Factory.setValidating( true);
DocumentBuilder builder = factory.newDocumentBuilder();

//set error handler for validating errors
Builder.setErrorhandler(new MyErrorHandler());

//obtain document object from XML document
Document =builder.parse(new File(file));
processNode(document);
}

Catch (SAXParseException spe){
System.err.println("Parse error: " +spe.getMessage());
System.exit(1);
}
Catch(SAXException se ) {
 Se.printStackTrace();
}
Catch(FileNotFoundException fne){
System.err.println("File\'"+file+"\'not found.");
System.exit(1);
}
Catch(Exception e){
e.printStackTrace();
}
}
Public void processNode(Node currentNode)
{
Switch(currentNode.getNodeType()){
//process a document node
Case Node.DOCUMENT_NODE:
Document doc=(Document ) currentNode;
System.out.println("document node:"+doc.getnodeName()+"\nRoot Element:"+
```

```
Doc.getDocumentElement().getNodeName());
processChildNodes(doc.getChildNodes());
break;
//process an element node
Case Node.ELEMENT_NODE:
System.out.orintln("\nElekent node:"+currentNode.getNodeName());
NameNodeMap attributeNodes=currentNode.getAttributes();
For(int i=0;i<atttributNodes.getLength();i++)
Attr attribute =(attr)attributeNodes.item(i);
System.out.println("\tAttributes:"+attributes.getNodeName()+";value="+
Attribute.getNodeValue());
}
processChildNodes(currentNode.getChildNodes());
break;
//process a text node and a CDATA section
Case Node.CDATA_SECTION_NODE:
Case Node.TEXT_NODE:
Text text = (text) currentNode;
If(!text.getNodeValue().trim().equals(""))
System.out.println("\tText:"+text.getNodeValue());
Break;
}}
Public void processChildNodes(NodeList children)
{
If(children.getLength()!=0)
For(int i=0;i<children.getLength();i++)
processNode(children.item(i));
}
public static void main(String args[])
{
If(args.length<1){
System.err.println(
"usage:java traversedom<filename>");
System.exit(1);
}
TraverseDOM traversedom =new TraverseDOM (args[0]);
}
}
```

Case Node.DOCUMENT_NODE:

Matches a document node. This case outputs the document node and process its child nodes by calling method processchildnodes

Case node.ELEMENT_NODE

Matches an element node. This case outputs the element's attributes and then processes its child nodes in process child nodes.

Java TraverseDOM simpleContact.xm;

Where java is the java interpreter, traverseDOM is the program being executed and simplecontact.xml is the command-line argument stored in array args. More precisely, args[0] contains the string simplecontact.xml

SIMPLE API FOR XML (SAX)

DOM VS. SAX

SAX and DOM are dramatically different APIs for accessing information in XML documents. DOM is a tree based model that stores the document's data in a hierarchy of nodes. Because all the document's data is in memory, data can be quickly accessed. DOM also provides facilities for adding or removing nodes.

SAX based parser invoke methods when markup is encountered. With this event based mode, no tree structure is created by the SAX based parser to store the XML document's data is passed to the application from the XML document as it is found. This results in greater performance and less memory overhead that with the DOM. In fact, many DOM parsers use a SAX parser to retrieve data from document for building the DOM tree. However, many programmers find it easier to traverse and manipulate XML documents using the DOM tree Structure. As a result, SAX parsers are typically used fro reading XML documents that will not be modified.

SAX-BASED PARSERS

SAX based parsers are available for a variety of programming language. Several SAX based parsers are available for free download. We use Sun Microsystem's JAXP with its default parser for the majority of the examples.

POSSIBLE QUESTIONS

PART B QUESTIONS
(EACH QUESTION CARRIES SIX MARKS)

1. Discuss the different types of nodes in DOM with their hierarchy.
2. Explain the advantages and Limitations of SAX.
3. Write a java program to traverse the DOM.
4. Explain DOM and SAX? Differentiate between them.
5. Explain the specialized exceptions for SAX Parsing. How are they handled?
6. Explain how to build an XML using DOM and Java programming.

7.Explain how to traverse an XML Document using Java Script.

8. Explain the methods of 'Node' Interface used in DOM Applications.
9. Explain the methods of 'Document' interface used in DOM.
10. b) Tabulate the events invoked by SAX parsers while processing an XML Document.

PART C QUESTIONS
(EACH QUESTION CARRIES TEN MARKS)

1. Explain the process and with suitable examples i) XLink ii) XPointer
2. Explain about the DOM Traverse Node.
3. Write a program to storing XML Files on the server and explain in detail.
4. Write a Java program to parsing an XML file using SAX.
5. Write a program to view an XML Student details and display the Student Details with an XSLT Stylesheet.

Karpagam Academy of Higher Education

Department of Computer Applications

Part-A Questions

Subject Code : 17CAP404W **Subject : XML** **Class : II MCA**

UNIT - III							
S.No.	**Questions**	**Choice 1**	**Choice 2**	**Choice 3**	**Choice 4**		**Answer**
1	Which of these is to manipulate the contents of the xml	XSL	DTD	CSS	DOM		DOM
2	W3C provides a standard recommendation to build the tree structure in the memeory called _____	XSL	DTD	CSS	DOM		DOM
3	DOM based parsers exposes a programatic library called	DOM-Application Programming Interface	Simple Application programming Interface for XML	Simple Application programming Interface for DOM	High Level Programming Interface		DOM-Application Programming Interface
4	DOM Interfaces are _____	Low Level Programming Interface	Platform independent	Platform dependent	Java Programs		Platform independent
5	_____ is a java based DOM Application Interface	JAPX	Jscript	JDOM	J#		JDOM
6	In the DOM tree the elements, attributes, contents etc., are all represented as _____	tags	nodes	branches	methods		nodes
7	The DOM tree is constructed in the _____	xml file	application program	computers memory	System Cache		computers memory
8	IBM's XML parser for Java is _____	JAPX	XML4J	JDOM	J#		XML4J
9	var xmldoc= new ActiveXObject("_____");	Microsoft.XMLDOC	Microsoft.XMLDOM	ActiveXObject.XMLDOM	XML.DOM		Microsoft.XMLDOM
10	In Javascript_____ method is called to get the xml document into the memeory	get( )	post( )	load( )	parse( )		load( )
11	Property_____ corresponds to the document's root element.	documentRoot	documentElement	documentFirstchild	documentNode		documentElement
12	Property _____ corresponds to the name of an element, attribute etc.	documentNode	nodeList	documentName	nodeName		nodeName
13	_____property is used to get the number of child nodes of a particular node	length	childNodes	nodeList	getChildren		length
14	Which of these interface represents the XML documents top level node?	Node	Root	Head	Document		Document
15	Which of these do not derives from Node interface	Element	Attr	Text	CharacterData		Text
16	Which of these derives from CharacterData interface	Comment	Text	Comment & Text	Data		Comment & Text
17	CDATASection interface derives from _____	Comment	Text	CharacterData	Node		Text
18	Which of these is not a node type	TEXT_NODE	COMMENT_NODE	ENTITY_NODE	ELEMENT_NODE		ENTITY_NODE
19	Individual child nodes of an element is accessed using the _____ method	child( )	individual( )	node( )	item( )		item( )
20	Nodes at the same level in a document is called _____	siblings	sisters	children	neighbour		siblings
21	The nodeValue method returns the value of the _____	Element node	Attribute node	Text Node	Comment Node		Text Node

22	The Element nodes have a node value _____	text	numbers	any data type	null		null
23	_____ returns the node in the previous level in the heirarchy	parentNode()	firstChild	lastChild	previousChild		parentNode()
24	_____ interface represents an attribute node.	Attribute	Attr	AttList	Attrib		Attr
25	Which method of the Node interface is used to duplicate a node	addChild	duplicateNode	cloneNode	appendChild		cloneNode
26	Which method of the Node interface is used to add a child node	addChild	duplicateNode	cloneNode	appendChild		appendChild
27	Which method of the Node interface is used to delete a child node	replaceChild	removeChild	deleteNode	deleteChild		removeChild
28	_____method is used to create a new Document object.	createDocument	createDOM	newDocument	Document.newInstance()		newDocument
29	_____ method is used to create a comment	<!-- comment -->	createComment	newComment	comment		createComment
30	_____ method of the XmlDocument is used to output the xml file	write	output	display	print		write
31	_____ uses an event based model for parsing the xml document	DOM	SAX	XML	JAXP		SAX
32	SAX was developed by----------	Tim Bernes Lee and ARPA funded students	H.M.Dietel and team	members of XML-DEV mailing list	Charles Goldfarb, Edward Mosher abd Raymond Lorie		members of XML-DEV mailing list
33	In SAX, notifications called _____ is raised when the document is parsed.	nodes	methods	subroutines	events		events
34	DOM is a _____ model	tree based	event based	object based	procedure based		tree based
35	Which of these is true about DOM	Documents data is stored in a hierarchy of nodes	The DOM tree is stored in the memory	Data can be accessed quickly	All data could be stored and access easily		All data could be stored and access easily
36	Which of these is not true about SAX	Document's data is stored in the memory	Raises events when parsed	Invokes methods to handle the events	XML data is passed to the application as it is encountered.		Document's data is stored in the memory
37	SAX parsers invokes methods when _____ is encountered	EOF	Nodes	CDATA	Markup		Markup
38	In SAX2.0 HandlerBase class us replaced by _____	EventHandler	DocumentHandler	DefaultHandler	HandlerMethod		DefaultHandler
39	_____ package provides the SAX programmatic interfaces required by a SAX parser.	org.xml.parsers	org.xml.sax	javax.xml.parsers	javax.xml.sax		org.xml.sax
40	_____ package provides classes and instantiating DOM and SAX parsers.	org.xml.parsers	org.xml.sax	javax.xml.parsers	javax.xml.sax		javax.xml.parsers
41	Class HandlerBase implements _____ interfaces	9	1	4	5		5
42	Class HandlerBase implements _____ interface for handling external entities.	EventHandler	DocumentHandler	DefaultHandler	EntityResolver		EntityResolver
43	Class HandlerBase implements _____ interface for handling notations and unparsed entitites	ErrorHandler	DocumentHandler	DTDHandler	EntityResolver		DTDHandler
44	Error Handler is a/an _____ for handling errors.	class	interface	method	event		interface
45	____ interface is for handling parsing events.	EventHandler	DocumentHandler	DTDHandler	EntityResolver		DocumentHandler
46	Which of these is not an interface that is implemented by the class HandlerBase.	EventHandler	DocumentHandler	DTDHandler	EntityResolver		EventHandler
47	Which of these interfaces are implemented by the class HandlerBase.	ErrorHandler	DTDHandler	EntityResolver	Entity and DTD Handler		Entity and DTD Handler
48	_____ method provides access to the parsed document's URL.	setDocumentLocator	getDocumentURL	getDocumentLocator	None of the above		setDocumentLocator

49	getSystemID method is called to retrieve a document's _____	SystemIP address	XML document's Root element	XML document's URL	None of the above		XML document's URL
50	_____ method is called when the document's root node is encountered.	startDocument	setDocumentLocator	BOF	getSystemID		startDocument
51	startElement method is called _____	when the root element is encounted	when a start tag is encountered.	when the BOF is encountered	WhenEOF is encountered.		when a start tag is encountered.
52	Which method is called last and only once	startDocument	endDocument	startElement	endElement		endDocument
53	What are the 2 arguments passed to method startElement?	Element name and type	Element name and its parent	Element name and its attributes	Element name and its child elements		Element name and its attributes
54	AttributList method getLength returns the ____	number of attributes the element has	size of the attribute	position of the last attribute	size of the attributes value		number of attributes the element has
55	In AttributeList the first attribute is at _____ position	0	1	Element position + 1	Parent position + 1		0
56	HandlerBase class member characters method is invoked when _____ is encountered	Text data	CDATA	comments	character data		character data
57	SAX parsers generate _____ when the XML document beign parsed is not well formed	fatal errors	validation errors	non fatal errors	warnings		fatal errors
58	SAX parsers generate warnings if the _____	XML document is not well formed	XML document is not valid	DTD is inconsistent	None of the above		DTD is inconsistent
59	SAX Parsers generates _____ while processing Invalidation documents.	fatal errors	nonfatal errors	warnings	no errors		nonfatal errors
60	_____ method of the HandlerBase class in invoked when the problem detected are not considered errors according to XML1.0	ErrorHandler	error	exception	warning		warning

UNIT-IV
SYLLABUS

XML Path Language: Nodes – Location Paths; XSLT: Templates - Creating Elements and attributes – Iteration and Sorting – Conditional Processing – Copying Nodes – Combining style sheets – variables. XLink, XPointer, XInclude and XBase.

XPATH

INTRODUCTION

- XML Path Language (XPath)
  - Syntax for locating information in XML document
    - e.g., attribute values
  - String-based language of expressions
    - Not structural language like XML
  - Used by other XML technologies
    - XSLT
    - XPointer

NODES

- XML document
  - Tree structure with nodes
  - Each node represents part of XML document
    - Seven types
      - Root
      - Element
      - Attribute
      - Text
      - Comment
      - Processing instruction
      - Namespace
    - Attributes and namespaces are not children of their parent node
      - They describe their parent node

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

2

3

<!-- Fig. 11.1 : simple.xml -->

4 <!-- Simple XML document -->

56

<book title = "C++ How to Program" edition = "3">
```

```
78
<sample>
9 <![CDATA[
10
11 // C++ comment
12 if ( this->getX() < 5 && value[ 0 ] != 3 )
13 cerr << this->displayError();
14 ]]>
15 </sample>
16
17 C++ How to Program by Deitel &amp;amp; Deitel
18 </book>
```

```
// C++ comment
if ( this->getX() < 5 && value[ 0 ] != 3 )
cerr << this->displayError();
```

The string-value for the sample element node is determined by concatenating the string-values for all of its *descendant text nodes* (i.e., all text nodes that follow the node) in *document order*. Nodes in an XPath tree have an ordering—called document order—that is determined by the order in which the nodes appear in the original XML document. The *reverse document order* is the reverse ordering of the nodes in a document. In this case, the sample element node has as its only descendent the text node on lines 9–14. Therefore, the string-value for the sample element node is the same as the string-value for the text node on lines 8–13. The book element node (lines 6–18) has two descendent text nodes. The first is the text node shown on lines 16 and 17. The second is the text node on lines 9–14. The string value for the book element node therefore consists of the concatenation of these two text nodes in document order. Thus, the resulting string-value is

```
// C++ comment
if ( this->getX() < 5 && value[ 0 ] != 3 )
cerr << this->displayError();
```

Because the text node on lines 16 and 17 is not contained within a CDATA section, it is normalized (i.e., whitespace is either removed or combined into a single whitespace character). For the root node of the document, the string-value is also determined by concatenating the string-values of its text-node descendents in document order. The string-value of the root node is therefore identical to the string-value calculated for the book element node. The attribute node title (line 6) has a string-value that consists of the normalized value of the attribute (i.e., C++ How to Program). The string-value for the edition attribute node consists of its value as well, which is 3. The string-value for a comment node consists only of the comment's text, excluding <!-- and -->. The string-value for the comment node on line 4 is therefore: simple XML document.

```
1 <?xml version = "1.0"?>
2
3
<!-- Fig. 11.3 : simple2.xml -->
4 <!-- Processing instructions and namespacess -->
5
6
<html xmlns = "http://www.w3.org/TR/REC-html40">
7
8
<head>
9 <title>Processing Instruction and Namespace Nodes</title>
10 </head>
11
12 <?deitelprocessor example = "fig11_03.xml"?>
13
14 <body>
15
16 <deitel:book deitel:edition = "1"
```

```
17 xmlns:deitel = "http://www.deitel.com/xmlhtp1">
18 <deitel:title>XML How to Program</deitel:title>
19 </deitel:book>
20
21 </body>
22
23 </html>
```

Fig. 1 XML document with processing-instruction and namespace nodes.

XPath node types.

Node Type	string-value	expanded-name	Description
root	Determined by concatenating the string-values of all text-node descendents in document order.	None.	Represents the root of an XML document. This node exists only at the top of the tree and may contain element, comment or processor-instruction children.
element	Determined by concatenating the string-values of all text-node descendents in document order.	The element tag, including the namespace prefix (if applicable).	Represents an XML element and may contain element, text, comment or processor-instruction children.
attribute	The normalized value of the attribute.	The name of the attribute, including the namespace prefix (if applicable).	Represents an attribute of an element.

LOCATION PATHS

- **Location path**
  - Expression specifying how to navigate XPath tree
  - Composed of *location steps*
    - Each location step composed of
      - Axis
      - Node test
      - Predicate

Axes

- XPath searches are made relative to *context node*
- Axis
  - Indicates which nodes are included in search
    - Relative to context node
  - Dictates node ordering in set
    - Forward axes select nodes that follow context node
    - Reverse axes select nodes that precede context node

XPath axes.

Name	Ordering	Description
	none	The context node itself.
nt	reverse	The context node's parent, if one exists.
d	forward	The context node's children, if they exist.
stor	reverse	The context node's ancestors, if they exist.
stor-or-self	reverse	The context node's ancestors and also itself.
endant	forward	The context node's descendants.
endant-or-self	forward	The context node's descendants and also itself.
owing	forward	The nodes in the XML document following the context node, not including descendants.
owing-sibling	forward	The sibling nodes following the context node.
eding	reverse	The nodes in the XML document preceding the context node, not including ancestors.
eding-sibling	reverse	The sibling nodes preceding the context node.
ibute	forward	The attribute nodes of the context node.
space	forward	The namespace nodes of the context node.

Node Tests

- Node tests
  - Refine set of nodes selected by axis
    - Rely upon axis' *principle node type*
      - Corresponds to type of node axis can select

Some XPath node tests

Node Test	Description
`*`	Selects all nodes of the same principal node type.
`node()`	Selects all nodes, regardless of their type.
`text()`	Selects all text nodes.
`comment()`	Selects all comment nodes.
`processing-instruction()`	Selects all processing-instruction nodes.
node name	Selects all nodes with the specified *node name*.

Location Paths Using Axes and Node Tests

- Location step
  - Axis and node test separated by double colon (**::**)
  - Optional *predicate* enclosed in square brackets (**[]**)
  - Some examples:
    - Select all element-node children of context node

child::*

- Select all text-node children of context node **child::text()**
- Select all text-node grandchildren of context node

child::*/child::text()

Some location-path abbreviations.

Location Path	Description

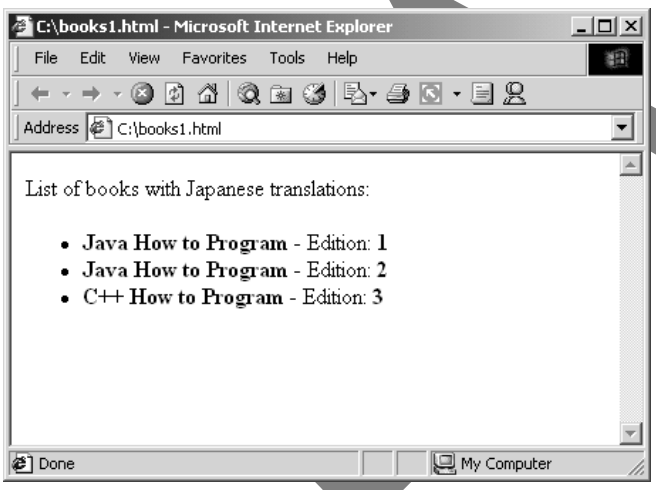
child::	This location path is used by default if no axis is supplied and may therefore be omitted.
attribute::	The attribute axis may be abbreviated as **@**.
/descendant-or-self::node()/	This location path is abbreviated as two slashes (//).
self::node()	The context node is abbreviated with a period (**.**).
parent::node()	The context node's parent is abbreviated with two periods (**..**).

XML document that marks up some book translations.

```
<?xml version = "1.0"?>
2
3
<!-- Fig. 11.9 : books.xml -->
4 <!-- XML book list -->
5
6
<books>
7
8
<book>
9 <title>Java How to Program</title>
10 <translation edition = "1">Spanish</translation>
11 <translation edition = "1">Chinese</translation>
12 <translation edition = "1">Japanese</translation>
13 <translation edition = "2">French</translation>
14 <translation edition = "2">Japanese</translation>
```

```
15 </book>
16
17 <book>
18 <title>C++ How to Program</title>
19 <translation edition = "1">Korean</translation>
20 <translation edition = "2">French</translation>
21 <translation edition = "2">Spanish</translation>
22 <translation edition = "3">Italian</translation>
23 <translation edition = "3">Japanese</translation>
24 </book>
25
26 </books>
```

Output



Location Paths Using Axes and Node Tests (cont.)

- Which books have Japanese translations?
  - Use root node of XPath tree as context node
  - Use predicate
    - Boolean expression for filtering nodes from search
    - Compare string value of current node to string '**Japanese**'

  /books/book/translation[. = 'Japanese']/../title

Node-set Operators and Functions

- Node-set operators
  - Manipulate node sets to form others
- Node-set functions
  - Perform actions on node-sets returned by location paths

Node-set operators.

Node-set Operators	Description
pipe (`|`)	Performs the union of two node-sets.
slash (`/`)	Separates location steps.
double-slash (`//`)	Abbreviation for the location path `/descendant-or-self::node()/`

Some node-set functions.

Node-set Functions	Description
`last()`	Returns the number of nodes in the node-set.
`position()`	Returns the position number of the current node in the node-set being tested.
`count(` *node-set* `)`	Returns the number of nodes in *node-set*.
`id(` *string* `)`	Returns the element node whose ID attribute matches the value specified by argument *string*.
`local-name(` *node-set* `)`	Returns the local part of the expanded-name for the first node in *node-set*.
`namespace-uri(` *node-set* `)`	Returns the namespace URI of the expanded-name for the first node in *node-set*.
`name(` *node-set* `)`	Returns the qualified name for the first node in *node-set*.

Node-set Operators and Functions

- Location-path expressions
  - Combine node-set operators and functions
    - Select all **head** and **body** children element nodes

head | body

- Select last **bold** element node in **head** element node

head/title[last()]

- Select third book element

book[position() = 3]

  - Or alternatively

book[3]

- Return total number of element-node children

count(*)

- Select all book element nodes in document

//book

XSLT

Introduction

The last chapter discussed XML Path Language (XPath) for locating specific nodes in an

XML document. *Extensible Stylesheet Language (XSL)* is used to format XML documents and consists of two parts. This chapter presents the first part of XSL—the *XSL Transformation Language (XSLT)*. XSLT transforms an XML document from one form to another. XSLT uses XPath to match nodes for transforming an XML document into a different document. The resulting document may be XML, HTML, plain text or any other text-based document. The second part of XSL is XSL *formatting objects*, which provide an alternative to CSS for formatting and styling an XML document. The following sections describe the use of XSLT and present many code examples.

Templates

An XSLT document is an XML document with a root element ***stylesheet***. The namespace for an XSLT document is **http://www.w3.org/1999/XSL/Transform**. The XSLT document shown in Fig. 12.1 transforms **intro.xml** (Fig. 12.2) into a simple HTML document (Fig 12.3). XSLT uses XPath expressions (discussed in Chapter 11) to locate nodes in an XML document. In an XSL transformation, there are two trees of nodes. The first node tree is the *source tree*. The nodes in this tree correspond to the original XML document to which the transformation is applied. The second node tree is the *result tree*. The result tree contains all of the nodes produced by the XSL transformation. This result

tree represents the document produced by the transformation. Lines 6 and 7 <xsl:stylesheet version = "1.0" xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"> show the XSLT document's root element (i.e., *xsl:stylesheet*) and its attributes. Attribute version defines the XSLT specification used. Namespace prefix *xsl* is defined and assigned the URI

"http://www.w3.org/1999/XSL/Transform". Line 9 <xsl:template match = "myMessage"> shows a *template element*. This element matches specific XML document nodes by using an XPath expression in attribute *match*. In this case, we match any myMessage element nodes.

```
<html>
<body><xsl:value-of select = "message"/></body>
</html>
1 <?xml version = "1.0"?>
23
<!-- Fig. 12.1 : intro.xsl -->
4 <!-- Simple XSLT document for intro.xml -->
5 6
<xsl:stylesheet version = "1.0"
7 xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
8
9
<xsl:template match = "myMessage">
10 <html>
11 <body><xsl:value-of select = "message"/></body>
12 </html>
13 </xsl:template>
14
15 </xsl:stylesheet>
```

Fig. 12.1 Simple template.

Lines 10–12 are the contents of the **template** element in line 9. When a **myMessage** element node is matched in the source tree, the contents of the **template** element are placed in the result

tree. By using element ***value-of*** and an XPath expression in attribute ***select***, the text contents of the node-set returned by the XPath expression are placed in the result tree. We will be using Internet Explorer 5 to process the XML and XSLT documents. By viewing the XML document, IE5 will automatically apply the XSLT document. Attribute **type** defines the type of file being attached. The two valid values are **text/xsl**, which denotes an XSL document, and **text/css**, which denotes a CSS document. Attribute **href** holds the file being attached.

Creating Elements and Attributes

In the previous section, we demonstrated the use of XSLT for simple element matching.

This section discuss the creation of new elements and attributes within an XSLT document.

```
1 <?xml version = "1.0"?>
23
<!-- Fig. 12.2 : intro.xml -->
4 <!-- Simple introduction to XML markup -->
56
<?xml:stylesheet type = "text/xsl" href = "intro.xsl"?>
78
<myMessage>
9 <message>Welcome to XSLT!</message>
10 </myMessage>
```

Fig. 12.2 Input XML document.

```
1 <html><body>Welcome to XSLT!</body></html>
```

```
1 <?xml version = "1.0"?>
23
<!-- Fig. 12.4 : games.xml -->
4 <!-- Sports Database -->
56
<sports>
```

```
78
<game title = "cricket">
9 <id>243</id>
10
11 <para>
12 More popular among commonwealth nations.
13 </para>
14 </game>
15
16 <game title = "baseball">
17 <id>431</id>
18
19 <para>
20 More popular in America.
21 </para>
22 </game>
23
24 <game title = "soccer">
25 <id>123</id>
26
27 <para>
28 Most popular sport in the world.
29 </para>
30 </game>
31
32 </sports>
```

```
1 <?xml version = "1.0"?>
23
<!-- Fig. 12.5 : elements.xsl -->
4 <!-- Using xsl:element and xsl:attribute -->
56
<xsl:stylesheet version = "1.0"
7 xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
89
<xsl:template match = "/">
10 <xsl:apply-templates/>
11 </xsl:template>
12
13 <xsl:template match = "sports">
14 <sports>
15 <xsl:apply-templates/>
16 </sports>
17 </xsl:template>
18
19 <xsl:template match = "game">
20 <xsl:element name = "{@title}">
21
22 <xsl:attribute name = "id">
23 <xsl:value-of select = "id"/>
24 </xsl:attribute>
25
```

```
26 <comment>
27 <xsl:value-of select = "para"/>
28 </comment>
29
30 </xsl:element>
31 </xsl:template>
32
33 </xsl:stylesheet>
```

```
1 <?xml version = "1.0" encoding = "UTF-8"?>
2 <sports>
34
<cricket id = "243">
5 <comment>
6 More popular among commonwealth nations.
7 </comment>
8 </cricket>
9
10 <baseball id = "431">
11 <comment>
12 More popular in America.
13 </comment>
14 </baseball>
15
16 <soccer id = "123">
17 <comment>
```

18 Most popular sport in the world.

19 </comment>

20 </soccer>

21

22 </sports>

ITERATION AND SORTING

XSLT also allows for iteration through a node set returned by an XPath expression. The node set can also be sorted.

1 <?xml version = "1.0"?>

23

<!-- Fig. 12.8 : usage.xml -->

4 <!-- Usage of elements and attributes -->

56

<?xml:stylesheet type = "text/xsl" href = "usage.xsl"?>

78

<book isbn = "999-99999-9-X">

9 <title>Deitel's XML Primer</title>

10

11 <author>

12 <firstName>Paul</firstName>

13 <lastName>Deitel</lastName>

14 </author>

15

16 <chapters>

17 <preface num = "1" pages = "2">Welcome</preface>

```
18 <chapter num = "1" pages = "4">Easy XML</chapter>
19 <chapter num = "2" pages = "2">XML Elements?</chapter>
20 <appendix num = "1" pages = "9">Entities</appendix>
21 </chapters>
22
23 <media type = "CD"/>
24 </book>
```

```
1 <?xml version = "1.0"?>
23
<!-- Fig. 12.9 : usage.xsl -->
4 <!-- Transformation of Book information into HTML -->
56
<xsl:stylesheet version = "1.0"
7 xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
89
<xsl:template match = "/">
10 <html>
11 <xsl:apply-templates/>
12 </html>
13 </xsl:template>
14
15 <xsl:template match = "book">
16 <head>
17 <title>ISBN <xsl:value-of select = "@isbn"/> -
18 <xsl:value-of select = "title"/></title>
```

```
19 </head>
20
21 <body bgcolor = "white">
22 <h1><xsl:value-of select = "title"/></h1>
23
24 <h2>by <xsl:value-of select = "author/lastName"/>,
25 <xsl:value-of select = "author/firstName"/></h2>
26
27 <table border = "1">
28 <xsl:for-each select = "chapters/preface">
29 <xsl:sort select = "@num" order = "ascending"/>
30 <tr>
31 <td align = "right">
32 Preface <xsl:value-of select = "@num"/>
33 </td>
34
35 <td>
36 <xsl:value-of select = "."/> (
37 <xsl:value-of select = "@pages"/> pages )
38 </td>
39 </tr>
40 </xsl:for-each>
41
42 <xsl:for-each select = "chapters/chapter">
43 <xsl:sort select = "@num" order = "ascending"/>
44 <tr>
```

```
45 <td align = "right">
46 Chapter <xsl:value-of select = "@num"/>
47 </td>
48
49 <td>
50 <xsl:value-of select = "."/> (
51 <xsl:value-of select = "@pages"/> pages )
52 </td>
53 </tr>
54 </xsl:for-each>
55
56 <xsl:for-each select = "chapters/appendix">
57 <xsl:sort select = "@num" order = "ascending"/>
58 <tr>
59 <td align = "right">
60 Appendix <xsl:value-of select = "@num"/>
61 </td>
62
63 <td>
64 <xsl:value-of select = "."/> (
65 <xsl:value-of select = "@pages"/> pages )
66 </td>
67 </tr>
68 </xsl:for-each>
69 </table>
70 </body>
```

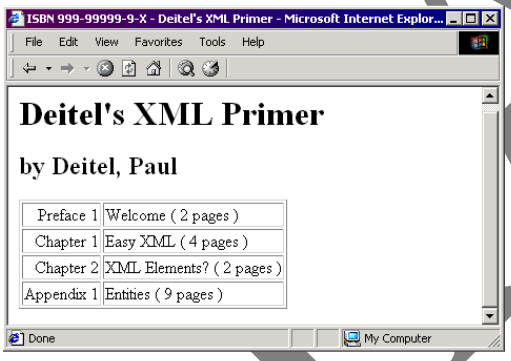
```
71 </xsl:template>
72
73 </xsl:stylesheet>
```

Fig. 12.9 Transforming XML data into HTML (part 3 of 3).

```
1 <html>
2 <head>
3 <title>ISBN 999-99999-9-X - Deitel's XML Primer</title>
4 </head>
56
<body bgcolor = "white">
7 <h1>Deitel's XML Primer</h1>
8 <h2>by Deitel, Paul</h2>
9
10 <table border = "1">
11 <tr>
12 <td align = "right">Preface 1</td>
13 <td>Welcome ( 2 pages )</td>
14 </tr>
15
16 <tr>
17 <td align = "right">Chapter 1</td>
18 <td>Easy XML ( 4 pages )</td>
19 </tr>
20
21 <tr>
```

```
22 <td align = "right">Chapter 2</td>
23 <td>XML Elements? ( 2 pages )</td>
24 </tr>
25
26 <tr>
27 <td align = "right">Appendix 1</td>
28 <td>Entities ( 9 pages )</td>
29 </tr>
30 </table>
31 </body>
32
33 </html>
```

Fig. 12.10 Output of the transformation.



CONDITIONAL PROCESSING

In the previous section, we discussed iteration of a node set. XSLT also provides elements to perform conditional processing, such as **if** statements.

```
1 <?xml version = "1.0"?>
23
```

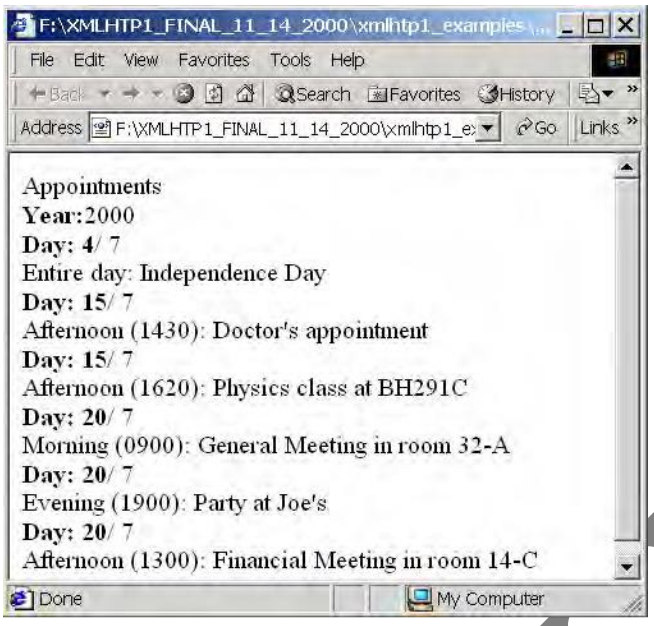
```
<!-- Fig. 12.11 : conditional.xsl -->
4 <!-- xsl:choose, xsl:when, and xsl:otherwise -->
56
<xsl:stylesheet version = "1.0"
7 xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
89
<xsl:template match = "/">
10 <html>
11
12 <body>
13 Appointments
14 <br/>
15 <xsl:apply-templates select = "planner/year"/>
16 </body>
17
18 </html>
19 </xsl:template>
20
21 <xsl:template match = "year">
22 <strong>Year:</strong>
23 <xsl:value-of select = "@value"/>
24 <br/>
25 <xsl:for-each select = "date/note">
26 <xsl:sort select = "../@day" order = "ascending"
27 data-type = "number"/>
28 <strong>
```

```
29 Day:
30 <xsl:value-of select = "../@day"/>/
31 <xsl:value-of select = "../@month"/>
32 </strong>
33
34 <xsl:choose>
35
36 <xsl:when test =
37 "@time &gt;= '0500' and @time &lt; '1200'">
38 Morning (<xsl:value-of select = "@time"/>):
39 </xsl:when>
40
41 <xsl:when test =
42 "@time &gt;= '1200' and @time &lt; '1700'">
43 Afternoon (<xsl:value-of select = "@time"/>):
44 </xsl:when>
45
```

Fig. 12.11 Using conditional elements (part 1 of 2).

```
46 <xsl:when test =
47 "@time &gt;= '1700' and @time &lt;= '2359'">
48 Evening (<xsl:value-of select = "@time"/>):
49 </xsl:when>
50
51 <xsl:when test =
52 "@time &gt;= '0100' and @time &lt; '0500'">
53 Night (<xsl:value-of select = "@time"/>):
```

```
54 </xsl:when>
55
56 <xsl:otherwise>
57 Entire day:
58 </xsl:otherwise>
59
60 </xsl:choose>
61
62 <xsl:value-of select = "."/>
63
64 <xsl:if test = ". = """>
65 n/a
66 </xsl:if>
67
68 <br/>
69 </xsl:for-each>
70
71 </xsl:template>
72
73 </xsl:stylesheet>
```



COPYING NODES

Instead of providing a template for each element of an XML document, XSLT provides an element to duplicate nodes from the source tree into the result tree. The XSLT element

copy is used to produce a copy of the context node and place it in the result tree.

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

23

<!-- Fig. 12.12: copyIntro.xsl -->

4 <!-- xsl:copy example using Intro.xml -->

56

<xsl:stylesheet version = "1.0"

7 xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">

89

<xsl:template match = "myMessage">

10

11 <xsl:copy>

12 <xsl:apply-templates/>
```

```
13 </xsl:copy>
14
```

Fig. 12.12 Using the XSLT element copy (part 1 of 2).

```
15 </xsl:template>
16
17 <xsl:template match = "message">
18
19 <xsl:copy>
20 How about &amp;apos;Hi World&amp;apos; for a change!
21 </xsl:copy>
22
23 </xsl:template>
24
25 </xsl:stylesheet>
```

Fig. 12.12 Using the XSLT element copy (part 2 of 2).

```
<?xml version = "1.0" encoding = "UTF-8"?>
2 <myMessage>
3 <message>
4 How about 'Hi World' for a change!
5 </message>
6 </myMessage>
```

Fig. 12.13 Resulting transformation

```
1 <?xml version = "1.0"?>
23
```

```
<!-- Fig. 12.14 : usingCopyOf.xsl -->
4 <!-- xsl:copy-of example using intro.xml -->
56
<xsl:stylesheet version = "1.0"
7 xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
8
9 <xsl:template match = "myMessage">
10
11 <xsl:comment>
12 The following XML tree has been copied into output.
13 </xsl:comment>
14
15 <xsl:copy-of select = "."/>
16 </xsl:template>
17
18 </xsl:stylesheet>
```

COMBINING STYLESHEETS

XSLT allows for modularity in stylesheets. This feature enables XSLT documents to import other XSLT documents.

```
1 <?xml version = "1.0"?>
23
<!-- Fig. 12.14 : usingCopyOf.xsl -->
4 <!-- xsl:copy-of example using intro.xml -->
56
<xsl:stylesheet version = "1.0"
```

```
7 xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
8
9 <xsl:template match = "myMessage">
10
11 <xsl:comment>
12 The following XML tree has been copied into output.
13 </xsl:comment>
14
15 <xsl:copy-of select = "."/>
16 </xsl:template>
17
18 </xsl:stylesheet>
```

Fig. 12.14 xsl:copy-of element.

```
1 <?xml version = "1.0" encoding = "UTF-8"?>
2 <!-- The following XML tree has been copied into output. -->
3 <myMessage>
4 <message>Welcome to XSLT!</message>
5 </myMessage>
```

provides a template to match any text and leftover element nodes.

```
1 <?xml version = "1.0"?>
23
<!-- Fig. 12.16 : usage2.xsl -->
4 <!-- xsl:import example -->
56
<xsl:stylesheet version = "1.0"
```

```
7 xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
89
<xsl:template match = "book">
10 <html>
11
12 <body>
13 <xsl:apply-templates/>
14 </body>
15 </html>
16
17 </xsl:template>
18
19 <xsl:template match = "title">
20 <xsl:value-of select = "."/>
21 </xsl:template>
22
23 <xsl:template match = "author">
24 <br/>
25
26 <p>Author:
27 <xsl:value-of select = "lastName"/>,
28 <xsl:value-of select = "firstName"/>
29 </p>
30
31 </xsl:template>
32
```

```
33 <xsl:template match = "*|text()"/>
34
35 </xsl:stylesheet>
```

Fig. 12.16 XSLT document being imported.

```
1 <?xml version = "1.0"?>
23
<!-- Fig. 12.17 : usage1.xsl -->
4 <!-- xsl:import example using usage.xml -->
56
<xsl:stylesheet version = "1.0"
7 xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
8
```

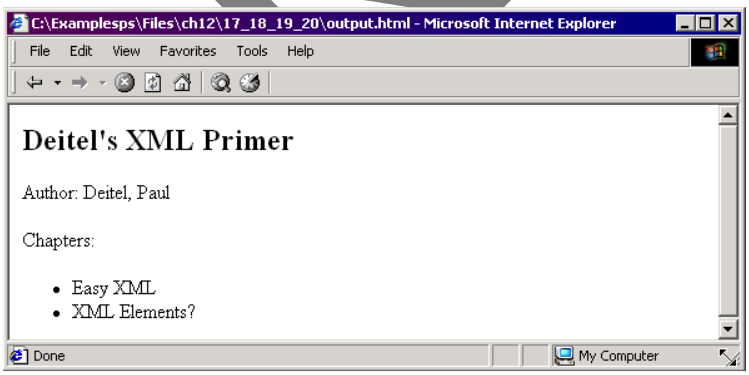
Fig. 12.17 Importing another XSLT document (part 1 of 2).

```
9 <xsl:import href = "usage2.xsl"/>
10
11 <!-- This template has higher precedence over the
12 templates being imported -->
13 <xsl:template match = "title">
14
15 <h2>
16 <xsl:value-of select = "."/>
17 </h2>
18
19 </xsl:template>
20
21 </xsl:stylesheet>
```

```
1 <html>
2 <body>
3 <h2>Deitel's XML Primer</h2>
4 <br>
5 <p>
6 Author: Deitel, Paul
7 </p>
8 </body>
9 </html>
```

```
1 <?xml version = "1.0"?>
23
<!-- Fig. 12.19 : book.xsl -->
4 <!-- xsl:include example using usage.xml -->
56
<xsl:stylesheet version = "1.0"
7 xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
89
<xsl:template match = "/">
10
11 <html>
12 <body>
13 <xsl:apply-templates select = "book"/>
14 </body>
15 </html>
```

```
16
17 </xsl:template>
18
19 <xsl:template match = "book">
20
21 <h2>
22 <xsl:value-of select = "title"/>
23 </h2>
24
25 <xsl:apply-templates/>
26 </xsl:template>
27
28 <xsl:include href = "author.xsl"/>
29 <xsl:include href = "chapters.xsl"/>
30
31 <xsl:template match = "*|text()"/>
32
33 </xsl:stylesheet>
```



Output of an XSLT document using element include (part 2 of 2).

VARIABLES

XSLT also provides the ability to keep variables for the processing of information. Figure 12.23 provides an example of an XSLT document using element **variable**. Lines 13 and 14 **<xsl:variable name = "sum" select = "sum(book/chapters/*/@pages)"/>** create element **variable** with attribute **name** of **sum**, for storing the sum of the number of pages in the book. Attribute **select** has value **sum(book/chapters/*/@pages)**, which is an XPath expression summing up the number of **pages** attributes of the elements in element **chapters**.

```
1 <?xml version = "1.0"?>
23
<!-- Fig. 12.23 : variables.xsl -->
4 <!-- using xsl:variables -->
56
<xsl:stylesheet version = "1.0"
7 xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
89
<xsl:template match = "/">
10
11 <total>
12 Number of pages =
13 <xsl:variable name = "sum"
14 select = "sum(book/chapters/*/@pages)"/>
15 <xsl:value-of select = "$sum"/>
16 </total>
```

```
17

18 </xsl:template>

19

20 </xsl:stylesheet>
```

Line 15 **<xsl:value-of select = "$sum"/>** uses element **value-of** to output the variable **sum** by using the dollar sign (**$**) to reference the variable. The **value-of** element can also be used to output the value of an element or attribute. Figure 12.24 shows the output of the XSLT document in Fig. 12.23 applied to the book XML document in Fig. 12.8. The XPath expression calculated the sum of the number of pages in each preface, chapter and appendix, which is 17 for the document in Fig. 12.8.

XLINK, XPOINTER, XINCLUDE AND XBASE

INTRODUCTION

XLink

(W3C Recommendation) –Describing links between resources (e.g., documents)

XPointer

(W3C Recommendation) –"Pointing"to document contents

XInclude

(W3C Recommendation) –Including existing XML document into another

XBase

(W3C Recommendation) –Specifies "base"URL for relative URLs XML Link Language (XLink)

XLINK

–Links to resources from XML documents

•files, images, documents, programs, and query results.

–Combined with XPointer, can address a portion of aresource, for example an element inside an XMLdocument.

XLink

Basic links and complex links

Allows XML documents to:

–Assert linking relationships among more than tworesources –Associate metadata with a link – Express links that reside in a location separate from the linked resources

To get access to XLinkfeatures the XLinknamespace must be declared:

–http://www.w3.org/1999/xlink

Simple Links

Simple link

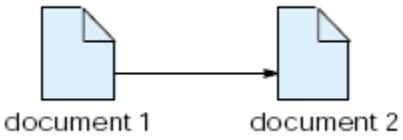
–Links one resource to another (similarly to HTML link) –Linking elements

•Specify linking information

<book xlink:type ="simple"xlink:href ="/textbooks/xmlHowToProgram.xml">XML Book </book>

• Linking element (book) islocal resource

• xml HowToProgram.xml is remote resource



XLinkAttributes

Attributes:

–

type: specifies the type of linking. The value must beone of "simple", "extended",

"locator", "arc","resource", "title", or "none".

href: defines the remote resource's URI

role: references a resource that describes the link (amachine readable description of the link)

title: a descriptive title for the link (human readable)

show: specifies how to display a resource when it isloaded

actuate:specifies when the resource should be retrieved

```
1<?xml version ="1.0"?>
23<!-- Fig. 14.4 : simpleLinks2.xml -->
4<!-- XML file that shows simple linking -->
56<contacts xmlns:xlink ="http://www.w3.org/1999/xlink">
78<contact
9xlink:type ="simple"
10xlink:href ="about.xml"
11xlink:role =http://www.deitel.com/xlink/contact
12xlink:title ="About Harvey Deitel"
13xlink:show ="new"
14xlink:actuate ="onRequest">
1516Dr. Harvey Deitel
17</contact>
1819</contacts>
```

2010.18

XLinkAttributes cont.

Attribute show:

–

new: indicates that the resource should be displayed in a newwindow or equivalent (depends on the application) –

replace: replaces the current resource with the linked resource –

embed: replaces the current element with the linked resource –

other or none allows the application to decide how to display thelink

Attribute actuate:

onRequest: resource should not be retrieved until the userrequests it –

onLoad: the document should be retrieved as soon as it isloaded –

other: application decides when to load the resource

Extended Links

Extended links

–Link multiple combinations of local and remoteresources

–Can create multidirectional links for traversingbetween resources.

–Not limited to just two resources, can link any numberof resources.

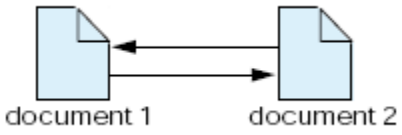
Unidirectional links

may not offer return to local resource•

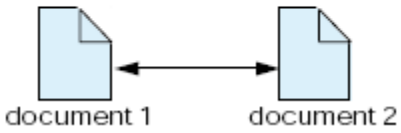
Multidirectional links

Traverse between resources

Unidirectional vsMultidirectional links

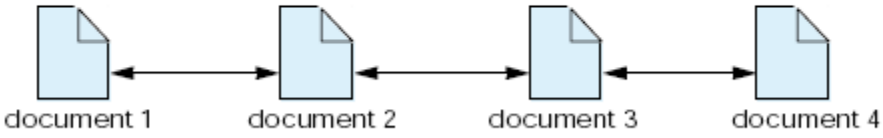


Two unidirectional links.



Multidirectional linking betweenfour resources.

Multidirectional link.




```
1 <?xml version = "1.0"?>
23
<!-- Fig. 14.8 : booklinks.xml -->
4 <!-- XML document containing extended links -->
56
<books xmlns:xlink = "http://www.w3.org/1999/xlink"
7 xlink:type = "extended"
8 xlink:title = "Book Inventory">
9
10 <author xlink:label = "authorDeitel"
11 xlink:type = "locator"
12 xlink:href = "authors/deitel.xml"
13 xlink:role = "http://deitel.com/xlink/author"
14 xlink:title = "Deitel &amp;amp; Associates, Inc.">
15 <persons id = "authors">
```

```
16 <person>Deitel, Harvey</person>
17 <person>Deitel, Paul</person>
18 </persons>
19 </author>
20
21 <publisher xlink:label = "publisherPrenticeHall"
22 xlink:type = "locator"
23 xlink:href = "/publisher/prenticehall.xml"
24 xlink:role = "http://deitel.com/xlink/publisher"
25 xlink:title = "Prentice Hall"/>
26
27 <warehouse xlink:label = "warehouseXYZ"
28 xlink:type = "locator"
29 xlink:href = "/warehouse/xyz.xml"
30 xlink:role = "http://deitel.com/xlink/warehouse"
31 xlink:title = "X.Y.Z. Books"/>
32
33 <book xlink:label = "JavaBook"
34 xlink:type = "resource"
35 xlink:role = "http://deitel.com/xlink/author"
36 xlink:title = "Textbook on Java">
37 Java How to Program: Third edition
38 </book>
39
40 <arcElement xlink:type = "arc"
41 xlink:from = "JavaBook"
```

```
42 xlink:arcrole = "http://deitel.com/xlink/info"
43 xlink:to = "authorDeitel"
44 xlink:show = "new"
45 xlink:actuate = "onRequest"
46 xlink:title = "About the author"/>
47
48 <arcElement xlink:type = "arc"
49 xlink:from = "JavaBook"
50 xlink:arcrole = "http://deitel.com/xlink/info"
51 xlink:to = "publisherPrenticeHall"
52 xlink:show = "new"
53 xlink:actuate = "onRequest"
54 xlink:title = "About the publisher"/>
55
56 <arcElement xlink:type = "arc"
57 xlink:from = "warehouseXYZ"
58 xlink:arcrole = "http://deitel.com/xlink/info"
59 xlink:to = "JavaBook"
60 xlink:show = "new"
61 xlink:actuate = "onRequest"
62 xlink:title = "Information about this book"/>
63
64 <arcElement xlink:type = "arc"
65 xlink:from = "publisherPrenticeHall"
66 xlink:arcrole = "http://deitel.com/xlink/stock"
67 xlink:to = "warehouseXYZ"
```

68 xlink:show = "embed"

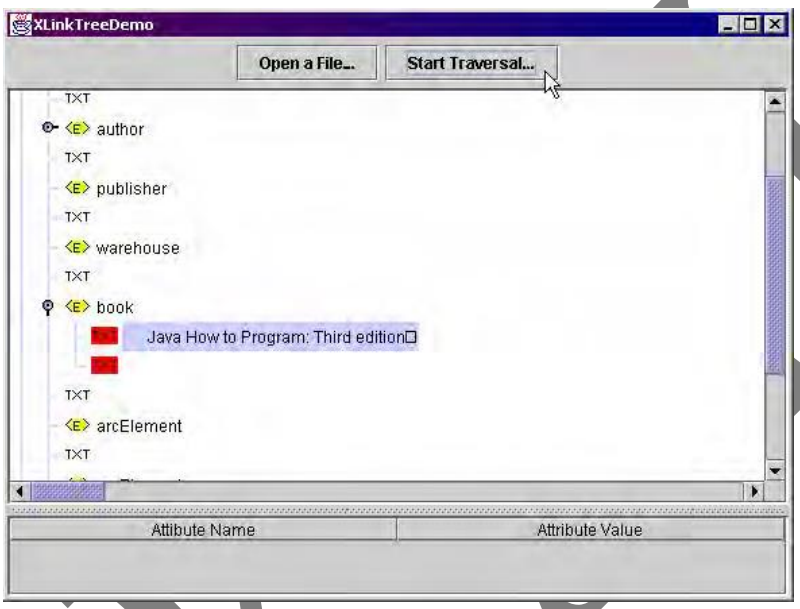
69 xlink:actuate = "onLoad"

70 xlink:title = "Publisher's inventory"/>

71

72 </books>

XML document containing extended links



XLink tree browser rendering of booklinks.xml

XML POINTER LANGUAGE (XPOINTER)

XPOINTER

–References fragments of XML document via URI

•Link to specific part of resource instead of linking to entireresource

•Link to specificlocations

(i.e., XPathtree nodes)•Link to ranges of locations

–Uses XPathto reference XML document nodes –Also used for searching XML documents via stringmatching

1<?xml version ="1.0"?>

2<!-- Fig. 14.14 : contacts.xml -->

3<!-- contact list document -->

45<contacts>

6<contact id ="author01">Deitel, Harvey</contact>

7<contact id ="author02">Deitel, Paul</contact>

8<contact id ="author03"> Nieto, Tem </contact>

9</contacts>

Mark up contact list that contains ids for three authors

–Assume contact list has relative URI/contacts.xml

• XLinkreferences entire contact list with URI

xlink:href ="/contacts.xml"

• XPointerreferences specific part:

xlink:href ="/contacts.xml#xpointer(//contact[@id =‘author02’])"

References element contact with id of author02 in document contacts.xml

XML INCLUSIONS (XINCLUDE)

Xinclude

–Reuse XML documents –Include XML documents within others

– U s e include element <includerxmlns:xinclude ="http://www.w3.org/1999/XML/xinclude"xinclude:href ="test.xml"xinclude:parse ="xml"/>

XML BASE (XBASE)

The W3C is currently working on a specification (currently a W3C Candidate Recommendation) to provide base URIs for relative links, similar to the HTML element **base**. XBase allows a

document author to change the base URI for relative links in a document. For the latest specification, visit **www.w3.org/TR/xmlbase**. In Section 14.4, we used the relative link **/contacts.xml** to reference a resource. If the base URI were **http://deitel.com**, then the complete URI for the resource would be **ttp://deitel.com/contacts.xml**. By using the XML attribute ***xml:base***, a document author can provide a new base URI.

For example,

<contact

xml:base = "http://deitel.net/"

xlink:type = "simple"

xlink:href = "/authors/author01biography.xml"

xlink:role = "http://deitel.com/xlink/contact"

xlink:title = "About this author">

uses attribute xml:base to provide the base URI http://deitel.net/ for attribute

href. The complete URI referenced by href is therefore

http://deitel.net/authors/author01biography.xml

POSSIBLE QUESTIONS

PART B QUESTIONS
(EACH QUESTION CARRIES SIX MARKS)

1. Explain the use of 'Location paths' to locate different parts of an XML Document.
2. Explain how to Combine Stylesheets in XSLT.
3. Explain the node types used in XPath.
4. Explain how the lists are formatted using XSLT.
5. Explain the Node-set operators and functions used in XPath.
6. Explain the syntax to sort elements using XSLT.
7. Explain about the use of Location paths and Axes in XPath.
8. Write a program including iteration and sorting techniques.
9. Explain the features of XPath Language and the Node types in it with example.
10. Explain with an example about Conditional Processing in XSLT.

PART C QUESTIONS
(EACH QUESTION CARRIES TEN MARKS)

1. Explain the process and with suitable examples i) XLink ii) XPointer
2. Explain about the DOM Traverse Node.
3. Write a program to storing XML Files on the server and explain in detail.
4. Write a Java program to parsing an XML file using SAX.
5. Write a program to view an XML Student details and display the Student Details with an XSLT Stylesheet.

Karpagam Academy of Higher Education

Department of Computer Applications

Part-A Questions

Subject Code : 17CAP404W **Subject : XML** **Class : II MCA**

UNIT - IV							
S.No.	**Questions**	**Choice 1**	**Choice 2**	**Choice 3**	**Choice 4**		**Answer**
1	In Xpath the XML document is conceptually viewed as a _____	text file	method	event	tree		tree
2	Nodes in the Xpath tree have an ordering called _____	node order	document order	node index	list index		document order
3	Each node except the root node has a _____	parent node	child node	attribute	node number		parent node
4	_____ is the reverse ordering of the nodes in a document.	document order	reverse document order	negative node order	negative node index		reverse document order
5	_____provides a syntax for locating specific parts of an XML document effectively and efficiently.	DOM	Schema	Xpath	XSL		Xpath
6	XPath is a _____ based language	object	event	structure	string		string
7	Which of these is not an Xpath node type	comment	entity	attribute	namespace		entity
8	Which if these node types cannot be a child node	element	processing instruction	text	namespace		namespace
9	_____ nodes has a parent node but they are considered as the child to that parent	element	processing instruction	text	namespace		namespace
10	Which of these node type cannot be a child node	root	attribute	namespace	root, attribute and namespace		root, attribute and namespace
11	Namespace nodes describes the namespace in which its _____ is found	parent node	child node	root node	element node		parent node
12	Each Xpath tree nodes has _____	attribute value	string value	node value	None of the above		string value
13	The string value of the text node consists of the _____	normalised attribute value	character data contined in the node	text within the CDATA section	normalized text with in the CDATA section		character data contined in the node
14	The _____ nodes has a string value that consists of the normalized attribute value	text	element	attribute	comment		attribute
15	The string value of the _____ node is determined by concatinating the string value of all of its descendent text nodes in document order	text	element	attribute	comment		element
16	Namespace nodes string value consists of _____	default namespace	a namespace prefix	URI of the namespace	sufix of the namespace		URI of the namespace
17	_____ consists of both the local part and a namespace URI.	Node name	Document order	Expanded name	String value		Expanded name
18	The Local part of the expanded name of a Processing Instruction node corresponds to its _____	target	value	node order	String value		target
19	The local part of the expanded name of a namespace node corresponds to its _____	default namespace	namespace prefix	empty	namespaceURI		namespace prefix
20	The namespace URI of the expanded name of a namespace node is _____	default namespace	namespace prefix	always null	Namespace		always null
21	_____ is an expression that specifies how to navigate an Xpath tree from one node to another.	Document order	String value	Location path	Expanded name		Location path

22	Searching through an XML document begins at a _____	root node	context node	Location path	Expanded name		context node
23	_____ indicates which node should be included in the search relative to the context node	Document order	String value	Node number	Axis		Axis
24	_____ dictates the ordering of the nodes in the set.	Document order	String value	Node number	Axis		Axis
25	Axes that select nodes that follow the context node in the document order is called _____	forward axes	backward axes	reverse axes	straight axes		forward axes
26	Axes that select nodes that precede the context node in the document order is called _____	forward axes	diagonal axes	reverse axes	straight axes		reverse axes
27	The set of nodes selected is refined with _____	node types	node tests	Node number	None of the above		node tests
28	An axis has a _____ that corresponds to the type of the node the axis select.	Xpath node type	Principle node type	String value	Expanded name		Principle node type
29	_____ is composed of a sequence of location steps.	Document order	String value	Location path	Expanded name		Location path
30	The location step consists of _____ seperated by double colon.	axis and node number	location path and axis	axis and node set	axis and node test		axis and node test
31	Node set _____ perform an action on a node-set returned by location path.	operators	tests	functions	None of the above		functions
32	Node set _____ allow us to manipulate the node-set	operators	tests	functions	None of the above		operators
33	_____ operator performs the union of two nodesets.	| (pipe)	/ (slash)	// (double slash)	:: (double colon)		| (pipe)
34	Node set funtion _____ returns the position number of the last node in the nodeset which is equal to the number of nodes in it.	count( )	last( )	Id( )	position( )		last( )
35	_____ funtion returns the number of nodes in the node set	count( )	last( )	both	none		both
36	Node set funtion name(ns) returns the qualified name of the ___ in the nodeset ns.	first node	node at current position	last node	nodeset itself		first node
37	Extensible Stylesheet Language is used to format XML documents and it conisists of _____ and _____	XSL, XSLT	XSL and Schema	XSLF,XSLT	XSLT and XSL Formatting objects		XSLT and XSL Formatting objects
38	Xpath expression is specified using _____	< >	[ ]	{ }	( )		{ }
39	_____ transforms XML document into other text based documents	Xpath	XSLT	XSLFormatting objects	Xlink		XSLT
40	Apache's Xalan is a _____	XSLT Processor	XML Parser	DTD validator	SAX Parser		XSLT Processor
41	The root element of the XSLT document is _____	ms:Schema	xsl:stylesheet	xmlns:transform	xsl:transform		xsl:stylesheet
42	_____ element matches specific XML document nodes by using an Xpath expression in the attribute match.	stylesheet	transform	variable	template		template
43	XSLT uses the element variable with an attribute _____ to define a variable.	type	value	order	name		name
44	In XSLT _____symbol is precedes the variable name to refer its value	@	$	#	?		$
45	_____XSLT element is used to output the value of an element or attribute	output	value	value-of	contents		value-of
46	Element _____in XSLt duplicates all children	clone	duplicate	copy-of	copy		copy-of

47	copy element in XSLT is used to copy only _____ nodes	element	attrbute	text	context		context
48	XSLT allows for modularity in stylesheets through _____ element to get other XSLT documents	import	module	match	select		import
49	Local templates have _____ imported templates.	higher precedence than	lower precedence than	equal precedence as	context based		higher precedence than
50	Local templates have _____ included templates.	higher precedence than	lower precedence than	equal precedence as	context based		equal precedence as
51	_____ is used when the result of the transformation of for print media.	XSLT	Formatting Objects	Structured document	Dom tree		Formatting Objects
52	Xlink is capable of linking _____	document	audio/vedio	database	all types of documents		all types of documents
53	Simple links are also called ____	hyperlinks	x-links	inline links	inline text		inline links
54	Xlink elements that specifies the linking information are called _____	hyperlinks	linking elements	inline links	x-links		linking elements
55	Xlink attribute _____ is to specify how to display a resource when it is loaded	show	view	display	actuate		show
56	Xlink attribute _____ is to specify when the resource should be retrieved.	show	view	retrieve	actuate		actuate
57	If attribute show is assigned a value _____ the linked resourse replaces the current element	new	replace	embed	other		embed
58	If attribute actuate is assigned a value onRequest the resourses is retrieved _____	as soon as it is loaded	when the user clicks the link	when the mouse points the link	None of the above		when the user clicks the link
59	Value of the attribute _____ is used to identify the resourse	label	name	type	id		label
60	Xlink attribute type is assigned a value locator for _____ resourses	local	extended	simple	remote		remote

UNIT-V
SYLLABUS

XML Technologies and Applications: XML Query Language – Directory Services Markup Language – Resources Definition Framework – XML topic Maps – Virtual Hyper Glossary – Channel Definition Format – Information and Content Exchange Protocol – Platform for Privacy preferences – XML Metadata Interchange.

XML TECHNOLOGIES AND APPLICATIONS

INTRODUCTION

Every day, individuals and organizations are discovering new and exciting ways to enhance existing technologies and create entirely new ones with XML. This chapter introduces several emerging XML-related technologies that cover a broad range of industries. In the first half of the chapter, we introduce the XML Query Language (XML Query), for searching and retrieving data from XML documents; Directory Services Markup Language (DSML), for describing relational data and metadata (i.e., information about information; elements are examples of metadata) so that they can be managed by directory services (e.g., software used to manage a company's personnel resources, etc.,); Resource Definition Framework (RDF), which enables document authors to describe the data in an XML document; an information-mapping technology called XML Topic Maps (STM); Channel Definition Format (CDF) and Rich Site Summary (RSS), which provide dynamic content to subscribers; and the Information and Content Exchange (ICE) Protocol, which manages content syndication over networks.

In the second half of this chapter, we introduce a specification for describing a Web site's privacy policy called the Platform for Privacy Preferences (P3P); a technology for transferring data over the Internet called the Blocks Extensible Exchange Protocol (BXXP), aW3C recommendation for XML implementation of security and authentication technologies called XML Digital Signatures; the Extensible Rights Markup Language (XrML), for licensing proprietary digital content and the XML Metadata Interchange (XML), for exchanging program-modeling data.

XML QUERY LANGUAGE (XML QUERY)

XML Query (XML Query Language) uses the power of XSL patterns to search XML documents for specific data. Just as SQL (Structured Query Language) searches for data stored in relational databases, XML Query searches for data stored in a XML document. XML Query syntax resembles the path specification in a UNIX environment (e.g., root/directory/subdirectory), XML Query was submitted to W3C as a proposal in 1998, and development on the language is ongoing.

DIRECTORY SERVICES MARKUP LANGUAGE (DSML)

Directory services provide a method for managing relational resources and metadata. A side from their usual for storing records of organizational assets, directory services can be used with XML, to dynamically match data across networks. The Directory Services Markup Language (DSML) is the bridge between directory services and XML. A standard vocabulary and schema provide the means for directory services information to be described in an XML document. With DSML, directories gain the ability to handle distributed Web-based applications, such as those used in e-business, network and supply chain management. DSML is platform independent, requiring only that the data be structured so that they can be manipulated with DSML.

URL/Description

www.w3.org/TandS/QL/QL98/pp/xql.html

This page contains the original XML Query proposal.

www.xml.com/pub/1999/03/quest/index2.html

This site contains a series of articles that summarize the results of the QL'98

Conference, which resulted in the submission of XML Query to the W3C.

metalab.unc.edu/xql

This site contains a brief XML Query FAQs.

Metalab.unc.edu/xql/xql-tutorial.html

This site contains a brief XML Query tutorial.

www.cuesoft.com/docs.cuexsl_activex/xql_users_guide.html

This site contains a detailed description of XML Query syntax.

www.w3.org/tR/xmluery-req

This site contains the W3C requirements for XML Query.

www.w3.org/TR/query-datamodel

This site defines the data model for XML Query.

www.dsml.org

This is the official Web site of the DSML standard.

www.oasis-open.org/cover/dsml.html

This site contains an introduction to DSML and links to relevant resources and articles.

RESOURCE DEFINITION FRAMEWORK (RDF)

The availability of the web and the relative ease of creating documents have led to a wealth of information on the web. Unfortunately, finding information on a specific topic can often be difficult and time consuming. The Resource definition Framework (RDF) is an XML- based language for describing information contained in a resource. A resource can be a web page, an entire Web site or any item on the Web that contains information in some form. RDF's "information about information" (or metadata) can be used by search engines or intelligent software agents to list or catalog information on the web. RDF can also be used to evaluate a Web site for rating purposes or to create digital signatures (i.e., the digital equivalent of a written signature). The Resource Definition Framework Model and Syntax is a Q3C Recommendation. The RDF Schema Specification Version 1.0 is currently a W3C Candidate Recommendation.

```
1   <?xml version = "1.0"?>
2
3   <! -- Fig 22.3 : simple.rdf -- >
4   <! -- Using RDF          -- >
5
6   <rdf : RDF
7       xmlns : rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
8       xmlns : dc = "http://purl.org/dc/elements/1.1/">
9
10      <rdf : Description about = "http://www.deitel.com">
11          <dc : Title>Deitel and Associates, Inc. </dc : Title>
12          <dc : Description>
```

```
13                  This is the home page of
14                  Deitel and Associates, Inc.
15                  </dc : Description>
16                  <dc : Date>2000-5-24</dc : Date>
17                  <dc : Format>text/html</dc : Format>
18                  <dc : Language>en</dc:Language>
19                  <dc : Creator>Deitel and Associates, Inc.</dc : creator>
20          </rdf : Description>
21
22      </rdf : RDF>
```

Fig. : 1 Simple RDF document describing a Web page.

Lines 6-8

```
<rdf : RDF
    xmlns : rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns : dc = "http://purl.org/dc/elements/1.1/">
```

define root element rdf : RDF. We declare the namespace profixes rdf and dc Namespace prefix rdf is used for RDF elements, and namespace prefix dc is used for metadata elements that are defined by the Dublin Core's Dublin Core Metadata Initiative. Dublin Core is an organization that is primarily concerned with metadata standards.

Line 10

```
<rdf : Description about = "http://www.deitel.com">
```

uses element rdf : Description to describe the resource specified in attribute about. In our case, we use the URL. http://www.deitel.com

Line 11

```
<dc : Title>Deitel and Associates, Inc. </dc : Title>
```

uses element Title to mark up the resource's name. Lines 12-20 use other metadata elements to provide further information about the resource.

Figure uses a visualization tool for RDF—located atwww.w3.org/RDF/Implementations/SiRPAC—to parse RDF documents into the RDF data model.

Figure presents a more substantial RDF document for describing the entire Deitel & Associates, Inc., Web site.

```
1       <xml version = "1.0"?>
2
3       <! -- Fig. 22.5 : links.rdf          -- >
4       <! – Describing entire Web site   -- >
5
6       <rdf : RDF xmlns:rdf = "http://www.w3.org/1999/02/22.-rdf-syntax-ns#"
7           xmlns : dc = " http://purl.org/dc/elements/1.1/">
8
9       <rdf : Description about = "www.deitel.com">
10        <dc: title>Home page of Deitel products </dc: Title>
11        <dc : Creator> Deitel and Associates, Inc.</dc : Creator>
12
13      <dc : subject>
14
15            <rdf : Bag ID = "links_1">
16            <rdf : li resource = "http://www.deitel.com/books/index.htm"/>
17              <rdf : li resource =
18              "http : //www.deitel.com/services/training/index.htm"/>
19            </rdf : Bag>
20
```

```
21          <rdf : Bag ID = “links_2”>
22             <rdf : li resource =
23                “http://www.deitel.com/announcements/contractors.htm”/>
24             <rdf : li resource =
25                “http://www.deitel.com/announcements/intership s.htm “/>
26          </rdf : Bag>
27
28          <rdf : Seq ID = “links_3”>
29                <rdf : li resource = “ http://www.deitel.com/intro.html “ />
30                <rdf : li resource = “http://wwww.deitel.com/directions.htm “/>
31          </rdf : Seq>
32
33          </dc:Subject>
34
35    </rdf : Description>
36
37    <! – description of the common feature of the Bag links_1 -- >
38    <rdf : Description aboutEach = “#links_1”>
39       <dc : Description>About our Products</dc : Description>
40    </rdf : Description>
41
42    <rdf :Description aboutEach = “links_2”>
43
44       <dc : Description>
45          Announcements, Opportunities
46          and Internships at Deitel and Associates, Inc.
```

```
47          </dc : Description>
48
49      </rdf : Description>
50
51      <rdf : Description aboutEach = “links_3 “>
52          <dc : Description> A;; about us </dc : Description>
53      </rdf : Description>
54
55      <! – further description of each link -- >
56      <rdf : Description about = “ http://www.deitel.com/books/index.htm “>
57
58
59      <! - - description of page title - - >
60      <rdf : title>
61          Books, Multimedia Cyber Classrooms
62          and complete Training Cources
63      </rdf : Title>
64
65      </rdf : Description>
66
67      <rdf : Description about =
68          “ http://www.deitel.com.services/training/index.htm “>
69          <rdf : Title>corporate Training Contractors </rdf : Title>
70      </rdf : Description>
71
72      <rdf : Description about =
```

```
73          " http://wwww.deitel.com/announcements/contractors.htm">
74          <rdf : Title> Looking for Training Contractors</rdf : Title>
75      </rdf : Description>
76
77      <rdf : Description about =
78          " http ://www.deitel.cim/announcements/internships.htm ">
79
80       <rdf : Title>
81          Ijnternships at deitel and Associates, Inc.
82      </rdf : Title>
83
84      <rdf : Description>
85
86      <rdf : Description about = " http://www.deitel.com.intro.htm ">
87
88          <rdf : Title>
89              Introduction to Deitel and Associates, Inc.
90          </rdf : Title>
91
92      </rdf :Description>
93
94      <rdf : Description about = " http://www.deitel.com/directions.htm ">
95          <rdf : title>Our location and how to get there</rdf : Title>
96      </rdf : Description>
97
98      </rdf : RDF>
```

Fig 2 RDF document describing an entire Web site (part 2 of 2)

Lines 15 and 21 use element rdf:Bag, which is a container element for an unordered list of resources. Element Bag contains li elements, which represent individual resources. Element seq is used (line 28) to represent an ordered list of resources.

By using attribute aboutEach with element Description, we provide a description of each resource. For example, lines 38-40 provide a description of each resource in the unordered list on lines 15-19. Figures 22.6 lists some RDF resources.

URL/Description

www.w3.org/RDF

This is the W3C's RDF information site.

www.w3.org/TR/REC-rdf-syntax-19990222

This is the RDF specification document.

www.oasis-open.org/cover/rdf.html

This site contains an introduction to RDF and links to resources and articles

www.ilrt.bris.sc.uk/discovery/rdf/resources

This site contains an extensive list of RDF resources.

www.w3.org/RDF/FAQ

This site contains RDF FAQs

www.wdv1.com/Authoring/Languages/RDF.html

This site contains a brief article on RDF, and includes links to other resources.

www.w3.org/DesignIssies/RDF-XML.html

This site is an article on the differences between RDF and XML.

Purl.oclc.org/dc

Home page of the Dublin Core Metadata Initiative.

XML TOPIC MAPS (XTM)

Topic maps, an International Standard Organization (ISO) standard, are a new model for navigating and linking resources. They can be thought of as a melding of an index, glossary, thesaurus and concept map (i.e., a graphical representation of data). Much as XML separates content from presentation, topic maps separate content form link. Topic maps, like XLink, exist as hypertext layers above, rather than within, an information set. They have many similar linking capabilities, including the ability to independently reference resources.

The topic the basic unit of a topic map is used to represent a subject and to reference data pertaining to that subject. Each topic is described by other topics that detail its names, occurrences and associations. The occurrences of a topic are the various information resources that relate to it. Each occurrence has a certain role, such as "chart" or "article," and a type that provides further information on the role. The various relationships between topic are described by using associations of different types as well as the role that each topic plays within the association. In addition, topic map incorporate mechanisms that define the identity and scope of a topic as well as the facets, or properties, of the information if references. Almost every component of a topic map is a topic, enabling the topic map to be self-documenting and self-describing. Indeed, processing instructions, queries and schemas for topic maps can themselves be expressed as topic maps. Depending on its application, a topic map may consider different subjects to be topics and deal with them in different ways. It logically follows that multiple topic maps may be applied to one information set and one topic map can be applied to multiple information sets.

```
1   <?xml version = "1.0"?>
2
3   <! – Fig. 22.7: vehicles.xml -- >
4   <! – simple topic map  -- >
5
6   <topicmap>
7
8     <! – topics -- >
```

```
<topic id = “light-vehicle”>

    <topname>
        <basename>Light weight vehicles</basename>
        <dispname>Passenger vehicles</dispname>
        <sortname>Light weight vehicles</sortname>
    </topname>

</topic>

<topic id = “heavy-vehicle”>

    <topname>
        <basename>Heavy weight vehicles </basename>
        <dispname>Construction vehicles</dispname>
        <sortname> Heavy weight vehicles</sortname>
    </topname>

</topic>

<topic id = “car” type = “light-vehicle”>

    <topname>
        <basename>ABC&amp;apos;s car</basename>
        <dispname>Sports car</dispname>
        <sortname>car</sortname>
```

```
35              </topname>
36
37              <! – occurrences of topic car -- >
38              <occurs occrl = “webpage” loctype = “URL”>
39                    http://www.adcsportscar.com/new
40              </occurs>
41
42              <occurs occrl = “order-online” loctype=”URL”>
43                    http://www.abcsportscar.con/outlet
44              </occurs>
45        </topic>
46
47        <topic id = “earth-mover” types = “heavy-vehicle”>
48
49            <topname>
50              <basename>ABC&amp;apos;s heavy duty mover</basename>
51              <dispname>Earth movers</dispname>
52              <sortname>Earth mover</sortname>
53            </topname>
54
55            <! – occurrences of topic earth-mover -- >
56            <occurs occrl = “webpage” loctype = “URL” >
57              http://www.abcsprotscar.com/heavy-vehicles
58            </occus>
59
60    </topic>
```

```
61
62      <topic id – “make”>
63          <topname>
64              <basename>ABC, Inc.</basename>
65          </topname>
66      </topic>
67
68      <topic id = “product”>
69           <topname>
70              <basename>Product</basename>
71           </topname>
72      </topic>
73
74      <! – associations -- >
75      <assoc type = “make”>
76          <assocrl type = “product”>car</assocrl>
77          <assocrl type = “product”>earth-mover</assocrl>
78      </assoc>
79      </topicmap>
```

```
c:\tmproc>c:\python\python –I tmproc.py vehicles.xml

Starting topic map processing….

Parsing vehicles.xml with xmlproc 0.62

Finished topic map processing….


>>>tm.get_topics()

[<tm.Topic light-vehicle>, <tm.Topic make>, <tm.Topic earth mover>, <tm.Topic car>, <tm.Topic product>, <tm.Topic heavy-vehicle>]


>>tm.get_types_list(tm[“car”])

[<tm.Topic light-vehicle>]


>>> tm.is_type(tm[“earth-mover”], tm[“heavy-vehicle”])

1


>>>tm.get_topic_of_type(tm[“heavy-vehicle”])

[<tm.Topic earth-mover”]


>>> topic = tm[“car”]

>>>topicname =  topic.get_names()[0]

>>>n =  topicname.get_basenames()[0]

>>>n.get_name()

“ABC’s car”


>>>dispname = topicname.get_dispnames()[0]

>>>dispname.get_name()
```

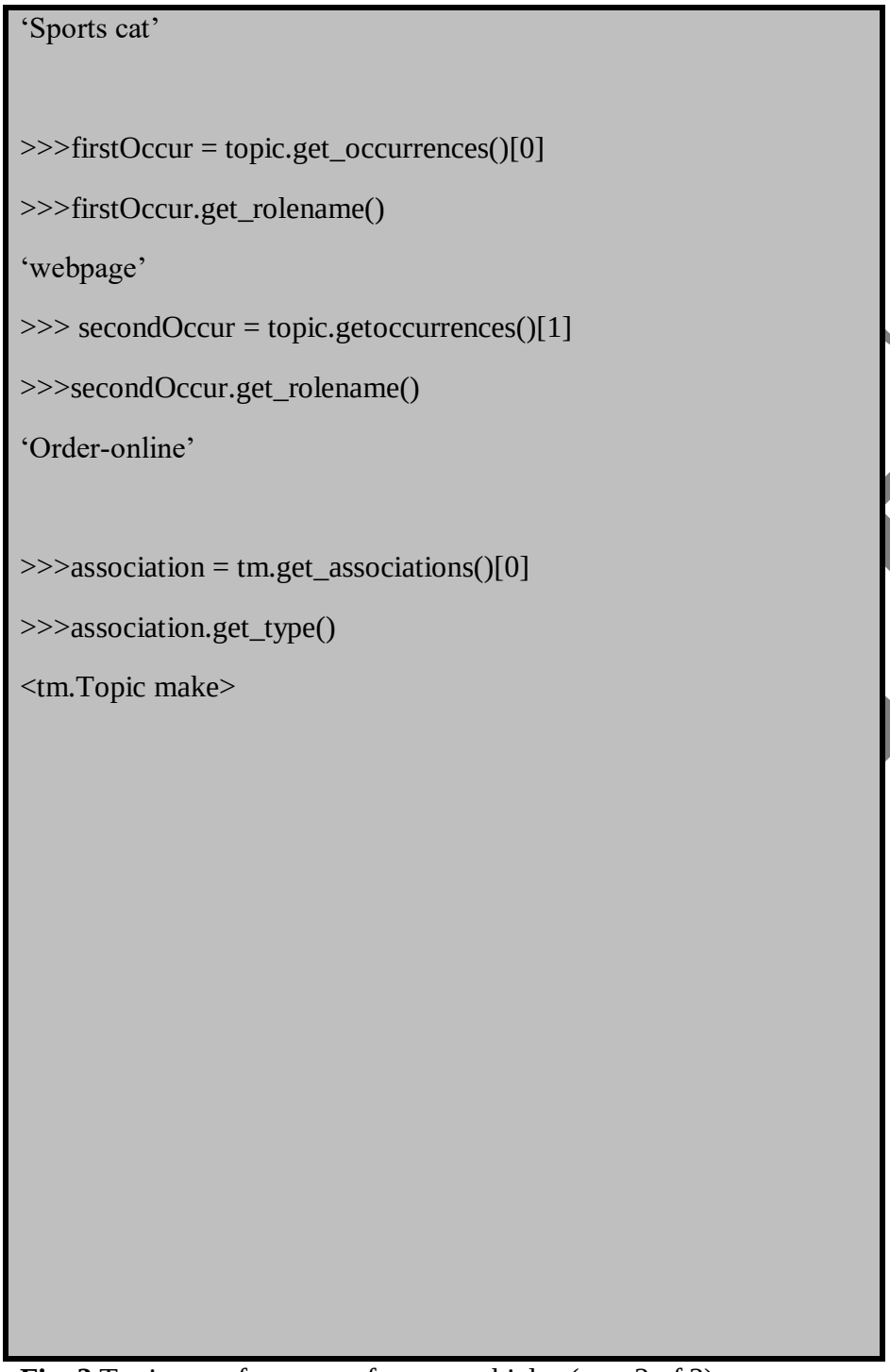


Fig. 3 Topic map for types of motor vehicles (part 3 of 3)

Currently, there are several preliminary XML topic-map implementations. For the example in this section, we use an implementation written in the Python programming language. To execute the example in Fig. 3, the following software must be installed.

1. Download Python Version 1.5.2 from **www.python.org/1.5** and follow the Installation instructions.
2. Download **saxlib** Version 1.0 (a Python version of SAX), driver package Version 1.01 and xmlproc (a validating XML parser) from the links is**www.stud.ifi.uio.no/-lmariusg/download/python/xml**.
3. Unpack all three packages into Lib subdirectory, e.g., c:\Python\Lib.
4. Download the topic-map implementation tmproc from **www.ontopicnet/software/tmproc/index.html**
5. Decompress tmproc to a local directory (e.g., c:\tmproc).

The Python implementation for topic maps can used to query the topic map. To run the application, at the command prompt (in tmproc subdirectory) enter the command.

C:\tmproc> c:\Python\python –i tmproc.py <file.xml>

This would start an interactive session with the user. Various commands (Fig. 22.8) in the tmproc library can be used to query the topic map.

As an ISO standard, topic maps have an SGML specification with a corresponding DTD, which been translated into several XML DTDs and schema. A standards committee is currently developing a specification for XML. Topic Maps (XTM).

.

VIRTUAL HYPER GLOSSARY (VHG)

VHG is a platform-independent specification for terminology. It provides a method of knowledge management that attaches extensive semantic information to structured data enabling, for instance, human-understandable definitions to be attaches to tags. VHG specifies a framework of hyperlinked glossaries, dictionaries and thesauri that provide documents with a terminological

environment. Documents can be linked to a glossary, which has an extensive network of internal hyperlinks. VHG takes advantage of the XML. DOM to gibes its glossaries a hierarchical structure that can be easily searched and indexed. In addition, VHG has software tools that give it advanced capabilities, including automatic lexical markup of documents, advanced indexing, concept-map generation, taxonomic manipulation of glossaries and support for multilingual glossaries using translation algorithms.

CHANNEL DEFINITION FORMAT (CDF)

CDF (Channel Definition Format) is a Microsoft XML application that implements a technology—called push technology—that automatically sends contents to users. Using CDF, Web authors can define channels that automatically deliver content to subscribed users. For example, a user can subscribe to a financial-information Web site's channel to receive information about certain stocks at a regular interval. Additionally, several related web sites can use channels to refer users between sites.

Methods	Description
get_topic	Returns all the topics found in the document.
get_types_list	List the types of a given topic.
is_type	Determines if a topic is of given type or not.
get_topics_of_type	Returns all the topics for a given type.
get_names	Returns all the names for a topic.
get_basenames	Returns the base names for a given topic.
get_dispnames	Returns the display names for a given topic.
get_occurrences	Returns the occurrence for a topic.
get_rolename	Return the role an occurrence.
get_associations	Returns all the associations in the document.
get_type	Returns the type of association or occurrence.

Fig 4 Some tmproc topic-map processor commands.

```
1.    <xml version = "1.0"?>
2.
3.    <! – Fig. 22.13 : simple.cdf    -- >
4.    <! – Simple example of CDF -- >
5
6.    <! DOCTYPE channel SYSTEM "cdf.dtd">
7.
8.    <channel href = "http://www/deitel.com">
9.
10        <title>Deitel Products </title>
11        <abstract> An example of CDF</abstract>
12
13        <logo href = http://www.deitel.com/images/logotiny.gif
14          style = "ICON" />
15
16        <item href =
17            http://www.deitel.com/books/1999/jhp3/jhtp3toc.htm>
18            <title>Java How to Program 3rd Edition </title>
19            <logo href =
20                http://www.deitel.com/images/0130125075_small.jpg
21                style = "ICON" />
22        </item>
23
```

```
24          <item href =
25              http://www.deitel.com/books/1999/w3htp/iw3htp_toc.htm>
26              <title> Internet and World Wide web How to Program </title>
27              <logo href =
28                  http://www.deitel/com/image/0130161438_small.jpg
29                      style = "ICON"
30          </item>
31
32      </channel>
```

Fig 5 CDF document for Deitel products.

Define a logo for the channel. Attribute href references the image, and attribute style has value ICON—indicating that the resource referenced by href is to be used as the icon for the channel.

Each item element provides a web page by using attribute href. Element item can also contain element title, abstract and logo.

For more information on Microsoft's CDF implementation, including additional elements and attributes, visit msdn.microsoft.com/workshop/delivery or the Web site.

INFORMATION AND CONTENT EXCHANGE (ICE) PROTOCOL

The Information and Content Exchange Protocol (ICE) was designed to facilitate the redistribution and reuse of web content. It automates the syndication, transfer and analysis of data, supporting a broad array of software systems and data formats. ICE governs the interaction between a media syndicator and a subscriber after they have entered into a business relationship. Communication with the protocol is based on the request/response model. The subscriber is sent a catalog of subscription offers, and ICE is used to establish and mange any subscription that are made. Data is contained in generic packages, which are transferred to the subscriber by one of several delivery models defined by ICE. A sequenced package model is available for both incremental and full updates, and their are also push and pull models. In addition to data transfer, ICE also provides a structure for exchanging and examining the event logs that keep track of the subscription. ICE also includes functionality for modifying the parameters of the protocol,

unsolicited text messaging and querying between the two systems. ICE has been submitted to the W3C or recommendation consideration. For additional information on ICE, consult the Web sites.

URL/Description

www.icestandard.org

This is the official site of the ICE standard

www.w3.org/TR/1998/NOTE-ice-19981026

This site contains the ICE document submitted to the W3C.

www.oasis-open.org/cover/ice.html

This site provides a brief introduction to ICE and provides links to resources and articles.

PLATFORM FOR PRIVACY PREFERENCES (P3P)

As Internet and e-commerce use continue to increase, users are becoming more aware of privacy issues. Many Web sites require visitors to divulge personal information in order to receive certain services. More ominously, companies such as Double Click have the potential to combine a user's personal information with the user's surfing and shopping preferences. In the last few years, federal and stat governments and independent organizations have raised several concerns about privacy.

The W3C is developing the Platform for Privacy Preferences (P3P) recommendation to help Web users manage how their personal information is collected and used on the Internet. A web site that uses P3P specifies what information the site request and how the site uses that information. The user's browser then interprets this information and compares the Information with user-defined privacy settings. For example the site may collect visitor information in order to send third-party information to the visitor's home address. If the visitor has specified that personal information should not be used for third-party offers, the visitor's browser could simply prevent access to the Web site or could display a message informing the visitor of the Web site's privacy policy and prompt the user for further action.

URL/Description

www.netscape.com/publish/help/mn20/quickstart.html

This is a Netscape documentation/tutorial site for RSS

Webreference.com/xml/column13

This site contains a how-to article on creating RSS content for a Web site

www.webreference.com/authoring/languages/xml/rss/intro/index.html

This site contains a technical introduction and prospects of RSS.

www.webreverence.com/authoring/languages/xml/rss

This site contains an extensive list of RSS resources.

My.netscape.com/publish/help/validate.tmpl

Users can validate their RSS documents at this site.

www.webreferences.com/perl/tools

This site lists available RSS tools for Perl

XML METADATA INTERCHANGE (XMI)

In a world of heterogeneous application, it is often difficult for developers to communicate and collaborate with each other. The Object Management Group (OMG) standardized the Unified Modeling Language (UML), giving developers a common language for designing object, distributed and business models for computing. UML is, in turn, part of a larger model, the Meta Object Facility (MOF), which describes all software modeling environment including itself. The MOF provides a rigorous definition of object-oriented models, technologies, semantics and data interchange formats. To enable the exchange of programming information across a network, UML and MOF were integrated with XMI, creating XMI, the OMG standard for sharing and storing object-oriented information. With XML, developers using different tools and programming in different development environments can collaborate and create compatible distributed applications. XMI is also a Stream-based Model Interchange Format (SMIF), which enables the streaming of object data from remote databases as well as from traditional file storage. There are standard XMI DTDs for many widely used object models, and it is possible to automatically generate DTDs for any meta-information model.

POSSIBLE QUESTIONS

PART B QUESTIONS
(EACH QUESTION CARRIES SIX MARKS)

1. Explain the purpose of Topic Maps with example.
2. Explain the purpose of XML Query with example.
3. Write a note on ICE and BXXP protocols.
4. Explain with an Illustration the application of RDF.
5. Explain the purpose of creating attribute groups.
6. Explain in detail about XML metadata Interchange (XMI).
7. Explain how to describe a web page using a simple RDF Document.
8. Explain the purpose of i) Virtual Hyper Glossary (VHG)
   ii) Rich Site Summary (RSS)
9. Explain the different types of names used for topics in XML topic maps.
10. Explain in which XML technologies are used to manage the privacy and security issues on the Web.

PART C QUESTIONS
(EACH QUESTION CARRIES TEN MARKS)

1. Explain the process and with suitable examples i) XLink ii) XPointer
2. Explain about the DOM Traverse Node.
3. Write a program to storing XML Files on the server and explain in detail.
4. Write a Java program to parsing an XML file using SAX.
5. Write a program to view an XML Student details and display the Student Details with an XSLT Stylesheet.

Karpagam Academy of Higher Education

Department of Computer Applications

Part-A Questions

Subject Code : 17CAP404W **Subject : XML** **Class : II MCA**

UNIT - V							
1	Which XML technology is used for searching and retrieving data from the XML document	XML Query Language	XML Topic Maps	XML Metadata Interchange	Virtual HyperGlossary		XML Query Language
2	P3P stands for _____	point to point protocol	platform for privacy preferences	Information and Content Exchage protocol	Privacy Protocol		platform for privacy preferences
3	Directories gain the ability to handle distributed Web applications like e-business through _____	RSS	XML Topic Maps	DSML	ICE Protocol		DSML
4	information about information is called _____	metadata	information description	data description	markup data		metadata
5	_____ is XML based language for describing information contained in a resource.	XML Topic Maps	XML Query Language	Resource Definition Framework	Channel Definition Format		Resource Definition Framework
6	BXXP is an acronym of	Block Extensible XML Protocol	Block Express Exchange Protocol	Block Exchange and XML Protocol	Blocks Extensible Exchange Protocol		Blocks Extensible Exchange Protocol
7	Search Engines use _____ to list or catalog inforamtion on the web.	Virtual HyperGlossary	RDF's metadata	XML Query Language	XML Topic Maps		RDF's metadata
8	Channel Definition Format is an XML application that uses _____ technology	push	pull	broadcast	packet switching		push
9	A CDF document has root element _____	CDF	format	channel	any		channel
10	_____ is an alternative to HTTP for transfering data over the internet.	FTP	P3P	ICE	BXXP		BXXP
11	RSS stands for	Religious Social Service	Rich Site Summary	Region Survey Summary	None of the above		Rich Site Summary
12	_____ implements push technology	Rich Site Summary	XML Topic Maps	XML Query Language	Resource Definition Framework		Rich Site Summary
13	In RDF documents the RDF elements uses the namespace prefix _____	rdf	xmlns	dc	XML Topic Maps		rdf
14	In RDF documents the metadata elements uses the namespace prefix _____	rdf	xmlns	dc	XML Topic Maps		dc
15	Element _____ in RDF is used to represent an ordered list of resources.	Ord	Bag	Seq	Res		Seq
16	Element _____ in RDF is a container of unordered list of resources.	unOrd	Bag	Seq	Res		Bag
17	_____ provides the document with a terminological environment	Virtual HyperGlossary	RDF's metadata	XML Query Language	XML Topic Maps		Virtual HyperGlossary
18	_____ specifies a framework of hyperlinked glossaries, dictionaries and thesauri	RSS	CDF	VHG	XTM		VHG

19	Communication with the ICE protocol is based on	push technology	pull technology	request/response model	packet switching		request/response model
20	ICE is a _____	protocol	language	edible	metadata		protocol
21	_____ governs the interaction between the media syndicator and a subscriber after they have entered into a business relationship	Ecom	ICE	P3P	BXXP		ICE
22	Information and Content Exchange Protocol is abbrevated as _____	(TCP) IP	BXXP	IXP	ICE Protocol		ICE Protocol
23	_____ is a security technology in XML	Public Key Cryptography	Encryption	Hashing Technology	XML Signature		XML Signature
24	The _____ describes all software modeling environment	Software Project Management	Meta Object Facility	Document Object Model	Software Modeling		Meta Object Facility
25	The ____ provides a rigorous definition of Object Oriented models,technologies and data interchange formats.		MOF	UML	OMG		MOF
26	UML and MOF are integrated with XML to create _____	UMI	OMG	XMI	DOM		XMI
27	_____ is an OMG standard for sharing and storing object oriented information	XML Metadata Interchange	Meta Object Facility	Document Object Model	None of the above		XML Metadata Interchange
28	_____ enables programmers to collaborate and create compatible distributed applications	XML Metadata Interchange	Meta Object Facility	Document Object Model	None of the above		XML Metadata Interchange
29	User must possess a _____ licence to access the e-content	XMI	XrML	XML signature	CDF		XrML
30	_____ provides a framework for legal agreements, certificationetc.	XMI	XrML	XML signature	CDF		XrML
31	XrML stands for Extensible rights Markup Language	Extensible rights Markup Language	Extensible Markup Language	XML Digital Signature	XML licence		Extensible rights Markup Language
32	The basic units of XML topic maps is _____	blocks	metadata	topic	element		topic
33	Each occurance of a topic has a role and _____ that provides information on the role.	name	type	scope	identity		type
34	Associations of topic map are represented by _____ element	associate	association	assoc	assistants		assoc
35	Occurance of a topic are described by _____ element	occurs	assoc	topic	basename		occurs
36	_____ gives the topic a name to be used by applications	topname	topicname	basename	dispname		basename
37	The root element of topic map is _____	topic	topicmap	topname	tmproc		topicmap
38	The root element of RDFdocument is _____	rdf:RDF	rdf:Root	rdf:Description	RDF		rdf:RDF
39	In _____ technology contents are sent automatically to the user.	push technology	pull technology	request/response model	packet switching		push technology
40	SQL Server provides how many Xquery methods ?	3	4	5	6		5
41	What is every identifier in an XQuery called ?	Name	QName	Qidt	Qnam		QName
42	Functions on numerical values in Xquery are :	ceiling	concat	contains	number		ceiling
43	Why do we use exist method in Xquery ?	To determine if the XML data contains a certain node	To examine the XML and return back a scalar value	To Shred the XML nodes of the XML data into relational columns	To search inside xml data types		To determine if the XML data contains a certain node
44	**Which of the following statements about XML schemas is incorrect?**	All XML documents must have a schema	Schemas can specify integer values	Schemas are defined by XSD tag	They offer more flexibility than DTDs		All XML documents must have a schema
45	**Which of the following statements is true:**	XML is a direct subset of SGML	SGML is an application of HTML	XML is a kind of dynamic HTML	XHTML is XML rewritten in HTML		XML is a direct subset of SGML

46	**The correct priority for implementing XML based IETMs is :**	Develop DTD, conduct a pilot project, create a modular library, train staff.	Train staff, convert legacy documents, develop DTD, create modular library.	Conduct pilot program, train staff, create modular library, develop DTD	Conduct pilot program, train staff, develop DTD, convert documents, purchace XML tools.		Conduct pilot program, train staff, create modular library, develop DTD
47	**What is an advantage of XML compared to HTML?**	XML works on more platforms.	XML is suited to using Web pages as front ends to databases.	XML was designed for portable phones.	XML is simpler to learn than HTML.		XML is suited to using Web pages as front ends to databases.
48	**Which of the following is a valid XSLT iteration command**	for	for-all	for-each	in-turn		for-each
49	**Which of the following XSLT Patterns is used to match the parent node**	/	//	.	..		..
50	**How can we make attributes have multiple values:**	<myElement myAttribute="value1 value2"/>	<myElement myAttribute="value1" myAttribute="value2"/>	<myElement myAttribute="value1, value2"/>	attributes cannot have multiple values		attributes cannot have multiple values
51	**The use of a DTD in XML development is:**	required when validating XML documents	no longer necessary after the XML editor has been customized	used to direct conversion using an XSLT processor	a good guide to populating a templates to be filled in when generating an XML document automatically		required when validating XML documents
52	**Parameter entities can appear in**	xml file	dtd file	xsl file	xml and dtd file		xml and dtd file
53	**Attribute standalone="no" should be included in XML declaration if a document:**	is linked to an external XSL stylesheet	has external general references	has processing instructions	has an external DTD		has an external DTD
54	In XML	the internal DTD subset is read before the external DTD	the external DTD subset is read before the internal DTD	there is no external type of DTD	there is no internal type of DTD		the internal DTD subset is read before the external DTD
55	**To use the external DTD we have the syntax**	<?xml version=”A.0” standalone=”no”?> <! DOCTYPE DOCUMENT SYSTEM “order.dtd”?>	<?xml version=”A.0” standalone=”yes”?> <! DOCTYPE DOCUMENT SYSTEM “order.dtd”?>	)<?xml version=”A.0” standalone=”no”?> <! DOCTYPE DOCUMENT “order.dtd”?>	<?xml version=”A.0” standalone=”yes”?> <! DOCTYPE DOCUMENT SYSTEM “order.dtd”?>		<?xml version=”A.0” standalone=”no”?> <! DOCTYPE DOCUMENT SYSTEM “order.dtd”?>
56	**To add the attribute named Type to the <customer> tag the syntax will be**	<customer attribute Type=”excellent”>	<customer Type attribute =”excellent”>	<customer Type attribute_type=”excellent”>	<customer Type=” excellent” >		<customer Type=” excellent” >
57	**Microsoft XML Schema Data types for Hexadecimal digits representating octates**	UID	UXID	UUID	XXID		UUID
58	**Microsoft XML Schema Data Type “ boolean” has values**	True ,False	True ,False or 1,0	1,0	any number other then zero and zero		1,0
59	**Simple type Built into Schema “ data’ represent a data in**	MM-DD-YY	Dd-MM-YY	YY-MM-DD	YYYY-MM-DD		YYYY-MM-DD
60	**The XML DOM object is**	Entity	Entity Reference	Comment Reference	Comment Data		Entity Reference

Reg. No.....

[12CAP505W]

KARPAGAM UNIVERSITY

(Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021
(For the candidates admitted from 2012 onwards)

MCA DEGREE EXAMINATION, NOVEMBER 2014

Fifth Semester

COMPUTER APPLICATIONS

XML

Time: 3 hours Maximum : 100 marks

PART – A (15 x 2 – 30 Marks)
Answer ALL the Questions

1. State any four rules for making an xml document.
2. What is XML parsing?
3. Write the syntax of defining an attribute in DTD.
4. How to define an anonymous custom simple type schema?
5. Write an xml document with mixed content element.
6. List any four numeric type definitions used in xml schema.
7. What is DOM?
8. Write syntax for creating element node in DOM.
9. Define SAX.
10. What is the use of XSLT?
11. Define XLink.
12. What is the use of location paths?
13. Define ICE.
14. State the use of XML.
15. What is VHG?

PART B (5 X 14= 70 Marks)
Answer ALL the Questions

16. a. What are the various types of attributes used in DTD? Explain with example.

Or

b. Explain in detail the types of elements in DTD with example.

17. a. Explain in detail the differences between DOM and SAX with example.

Or

b. Give an example to explain the components of DOM.

18. a. Discuss in detail the concept of XML Path Language.

Or

b. How to create elements with attributes and perform iteration and sorting on the elements in XSLT? Explain.

19. a. Write in detail about RDF and XML Topic Maps.

Or

b. Discuss CDF and XQL in detail.

20. **Compulsory : -**

Write a Complex type schema for an XML document is given below
<parcel_detail? <parcel_id>. </parcel_id> <date> ...</date> <weight> .. </weight> <charges> .... </charges> <sender><Name>...</Name> <address> ....</address> <phone> .....</phone> </sender> <receiver> .....</receiver> </parcel_detail>
Constraints : the element <parcel_id> should be an ID type element, the tag <receiver> will have the same set of elements that the <sender> tag has, and the element <phone> should accept only 10 positive integer digits.

Reg. No.....

[14CAP505W]

KARPAGAM UNIVERSITY

Karpagam Academy of Higher Education
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021
(For the candidates admitted from 2014 onwards)

MCA DEGREE EXAMINATION, NOVEMBER 2016

Fifth Semester

COMPUTER APPLICATIONS

XML

Time: 3 hours Maximum : 60 marks

PART – A (20 x 1 = 20 Marks) (30 Minutes)
(Question Nos. 1 to 20 Online Examinations)

PART B (5 x 8 = 40 Marks) (2 ½ Hours)
Answer ALL the Questions

21. a) Write the history of World Wide Web

Or

b) Explain the features and components of XML with Illustration.

22. a) Explain how to describe attributes in Msxml schema.

Or

b) Explain the purpose of Occurrence Indicators in DTD

23. a) Explain how to traverse an XML Document using Java Script

Or

b) Explain the methods of 'Node' interface used in DOM Applications

24. a) Write the Node-set operators and functions used in XPath

Or

b) Explain the syntax to sort elements using XSLT

25. a) Tabulate the tmproc topic map processor commands and their purposes

Or

b) What is XML metadata Interchange (XMI)
