CHAPTER 21

The Document Object Model

In This Chapter

Introducing the Document Object Model

Understanding the document hierarchy

Using the DOM Objects

Microsoft has developed a tool for creating and parsing XML documents called MSXML. MSXML implements the XML Document Object Model (DOM) as defined by the Worldwide Web Consortium (usually called W3C). W3C is an international standards body that is responsible for the standards (such as HTML, XML, XSP, and so on) that are used to access the Web.

The DOM stores an XML document as a series of objects that can be easily accessed by an application program. MSXML implements DOM as a series of COM objects for use on Windows-based computers. This means that these objects are independent of any programming language and database server and can be used with Visual Basic, Visual C++, ASP script files, SQL Server, and Oracle 8i without problems.

In this chapter, I'll discuss how to use the Document Object Model to create and parse XML documents.

Basically beta: The version of the Document Object Model discussed in this book is based on the March 2000 beta release of MSXML. At the time I wrote this chapter, the W3C hadn't finalized the specifications for DOM, and as a consequence, Microsoft's implementation of it in MSXML is still subject to change.

The Document Object Model

Microsoft's implementation of the Document Object Model conforms fairly closely to the W3C's recommendation, but recommendations can be fairly complex to understand. The DOM is based on a hierarchical organization of object nodes whose types vary, as different types of information are stored in the

document hierarchy. In addition, a number of other objects are stored there that contain specific pieces of information or utility functions.

Document hierarchy

The basic DOM hierarchy is shown in Figure 21-1. The root node of the hierarchy is the `DOMDocument` object, which encompasses the entire XML document. As you descend through the tree structure, other nodes represent a particular piece of information or group of information in the XML document.

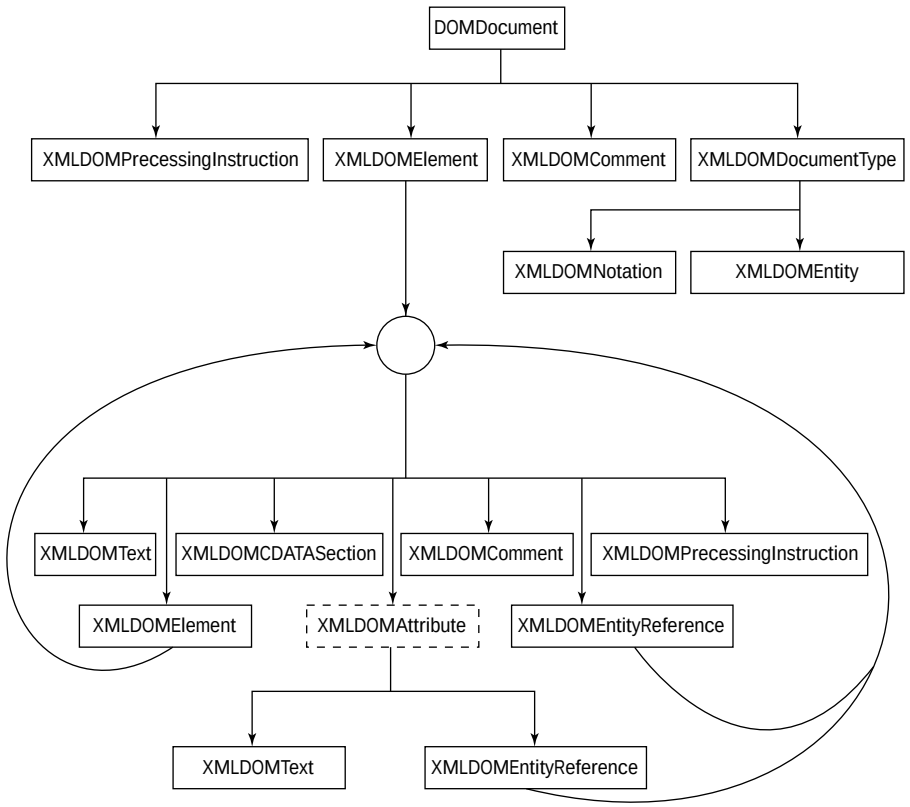

Figure 21-1: Viewing the Document Object Model object hierarchy

The basic object is the `XMLDOMNode` object, from which all of the other nodes are derived. In total, there are a dozen different types of nodes. Aside from the `DOMDocument` object, all of the other objects begin with `XMLDOM`.

- `DOMDocument` – This is the highest-level object in the hierarchy. It encompasses the entire XML document. It also includes additional methods and properties that can be used to obtain an XML document or to create other nodes.
- `XMLDOMAttribute` – This object is used to hold information about a single attribute in an element. It is not considered a true node value since it is referenced through the `attributes` property, rather than the `childNodes` property.
- `XMLDOMCDATASection` – This object is similar to the `IXMLDOMText` object, except it contains data that was stored in a CDATA section in the XML document. The CDATA section begins with the `<[[` string and ends with the `]]>` string. Any characters in between the two CDATA delimiters are treated as normal text. This lets you include any text into a text field, including elements that would normally be processed by the XML parser.
- `XMLDOMComment` – This object holds an XML comment element, which is the string of characters between the `<!--` and `-->`.
- `XMLDOMDocumentType` – This object contains information about the Document Type Declarations included in an XML document.
- `XMLDOMElement` – This object contains information about a single element in an XML document. If the element in the document contains other, nested elements, then the `childNodes` property will contain references to these objects. The `attributes` property can be used to find the `IXMLDOMAttribute` objects, which contain information about the attributes found in this element.
- `XMLDOMEntity` – This object contains information about a parsed or unparsed entity in the XML document.
- `XMLDOMEntityReference` – This object contains information about an entity reference in the XML document.
- `XMLDOMNode` – This object is the fundamental object of the Document Object Model. All of the other objects that can act as a node in the hierarchy are derived from this object.
- `XMLDOMNotation` – This object contains notation information from a Document Type Declaration or a schema.
- `XMLDOMProcessingInstruction` – This object contains processing information for the XML document. For example, this object will store the information found in the `<?xml version="1.0"?>` tag that is found at the start of most XML documents.
- `XMLDOMText` – This object contains the text value found in between the `<element>` and `</element>` tags in a document.

To IXMLDOM or not to IXMLDOM: Microsoft frequently prefixes the node names with an `I` in their implementation of the object model. The `I` prefix is a C++ convention that indicates that the object is a COM Interface, which just another object to a Visual Basic programmer.

Other objects

Here are a few other objects that you may find interesting. These objects cannot be used in place of a node object, but offer specialized support services.

- `XMLDOMDocumentFragment` – This object is used to store part of an XML document. You might find it useful if you want to restructure an XML object hierarchy.
- `XMLDOMImplementation` – This object allows you to detect various features included in a particular implementation of the Document Object Model.
- `XMLDOMNodeList` – This object is a container for a collection of nodes. This object is typically returned by several methods that you can use to search for specific elements in the document hierarchy.
- `XMLDOMParseError` – This object contains information about any errors that occurred while parsing your XML document.
- `XMLHttpRequest` – This object allows you to send and receive information using an HTTP connection.

The XMLDOMNode Object

The `XMLDOMNode` object is the fundamental object in the Document Object Model. It forms the basis of all the other node objects that hold your XML document. Each of those objects will inherit the `XMLDOMNode`'s properties and methods, to which it may add unique properties, methods, and events.

XMLDOMNode object properties

Table 21-1 lists the properties associated with the `XMLDOMNode` object.

Table 21-1
Properties of the XMLDOMNode object

Property	*Description*
`attributes`	An object reference to a node containing the attributes. Returns `NULL` if this node doesn't have any attributes.
`baseName`	A `String` value containing right side of a fully qualified name of the document.
`childNodes`	An object reference to one of several different collections depending on the value of `nodeType` (see Table 21-2).
`dataType`	A `Variant` value that is either a `String` containing the name of the data type associated with the node or `VT_NULL` if no data type is defined.
`definition`	An object reference to a node in the document type definition or schema, if included.
`firstChild`	An object reference to the first child node of this node. Returns `NULL` if this node doesn't have any children.
`lastChild`	An object reference to the last child node of this node. Returns `NULL` if this node doesn't have any children.
`namespaceURI`	A `String` containing the Universal Resource Identifier (URI) of the namespace.
`nextSibling`	An object reference to the next sibling of this node. If there are no more siblings, then this value is `NULL`.
`nodeName`	A `String` value containing the name of the node.
`nodeType`	An enumerated data type containing the type of the node (see Table 21-2).
`nodeTypedValue`	A `String` value containing the node's value in its defined data type.
`nodeTypeString`	A `String` value containing a value corresponding to the value of `nodeType`. See Table 21-2 for the string values that correspond to the `nodeType` value.
`nodeValue`	A `String` value containing the text associated with the node.
`ownerDocument`	An object reference to the root `DOMDocument` object or the `DOMDocument`, which created this object.

Continued

Table 21-1 *(continued)*

Property	*Description*
parentNode	An object reference to the parent object of this node.
parsed	A Boolean when True means that this node and all of its descendents have been parsed.
prefix	A String containing the namespace prefix.
previousSibling	An object reference to the previous sibling of this node. If this is the first node, then this value will be NULL.
specified	A Boolean value when True means that the value was specified, rather than being derived from a default value in the DTD or schema.
text	A String containing the text value of the current node and all of its subtrees.
xml	A String containing the XML statements that make up the current document.

XMLDOMNode object methods

The XMLDOMNode object has a number of methods that are used to manage the child nodes associated with it.

Function appendChild (newChild As IXMLDOMNode) as IXMLDOMNode

The appendChild method is used to add a node object as a child of the current node. It returns an object to the newly added child node. This method is the equivalent of calling insertBefore (newChild, NULL). newChild is an object reference to a node, which will be added as a child to the current node.

Function cloneNode (deep As Boolean) as IXMLDOMNode

The cloneNode method is used to create a copy of the current node. deep is a Boolean, when True means that all of the child nodes of the current node are to be cloned also.

Table 21-2
Values for nodeType

Constant	*Integer*	*String*	*Description*
NODE_ELEMENT	1	element	Represents an element. Can be a child of the Document, DocumentFragment, EntityReference and Element nodes. Can have one or more Element, Text, Comment, ProcessingInstruction, CDATASection, and EntityReference nodes as children.
NODE_ATTRIBUTE	2	attribute	Represents an attribute. Referenced though an Entity node. Can have the Text and the EntityReference nodes as children. Technically, an Attribute is not considered a child of the Element node, since it is referenced through the attributes property.
NODE_TEXT	3	Text	Represents the text content of a tag. Can be a child of the Attribute, DocumentFragment, Element, and EntityReference nodes. Cannot have any child nodes.
NODE_CDATA_SECTION	4	cdatasection	Represents a CDATA section, which is used to escape blocks of text that might be recognized as markup. Can be a child of the DocumentFragment, EntityReference, and Element nodes. Cannot have any child nodes.
NODE_ENTITY_REFERENCE	5	Entityreference	Represents an entity. Can be a child of the Attribute, DocumentFragment, Element, and EntityReference nodes. Can have the Element, ProcessingInstruction, Comment, Text, CDATASection, and EntityReference nodes as children.
NODE_ENTITY	6	Entity	Represents an expanded entity. Can be a child of the DocumentType node. Can have other nodes as child nodes.

Continued

Table 21-2 *(continued)*

Constant	*Integer*	*String*	*Description*
NODE_PROCESSING_INSTRUCTION	7	processinginstruction	Can be a child of the Document, Document Fragment, Element, and EntityReference nodes. Cannot have any child nodes.
NODE_COMMENT	8	comment	Represents a comment. Can be a child of the Document, DocumentFragment, Element, and EntityReference nodes. Cannot have any child nodes.
NODE_DOCUMENT	9	document	Represents an XML document. Cannot be a child of any other nodes. Can have exactly one child node, which can be an Element, a ProcessingInstruction, a Comment, or a DocumentType node.
NODE_DOCUMENT_TYPE	10	Documenttype	Represents a document type declaration. Can be a child of the Document node. Can have Notation and Entity nodes as child nodes.
NODE_DOCUMENT_FRAGMENT	11	documentfragment	Represents a node or subtree with a document without being contained within the document. Cannot be a child of any other nodes. Can have Element, ProcessingInstruction, Comment, Text, CDATASection, and EntityReference nodes as child nodes.
NODE_NOTATION	12	notation	Represents a notation in the document. Can be a child of the DocumentType node. Cannot have any child nodes.

Function hasChildNodes () as Boolean

The `hasChildNodes` method returns `True` if the current node has children.

Function insertBefore (newChild As ICMLDOMNode, refChild) as IXMLDOMNode

The `insertBefore` method is used to add a child node to the current node. The node will be inserted before the node specified in `refChild`. An object reference to `newChild` will be returned as the value of the method.

`newChild` is an object reference to the node to be added. `refChild` is an object reference to a node that is a child of the current node. If this parameter is `NULL`, then the node will be added to the end of the collection.

Function removeChild (childNode as ICMLDOMNode) as IXMLDOMNode

The `removeChild` method is used to remove the specified node from the current node's child nodes. The node is not destroyed, so you should set the node to `Nothing` to free the node's resources if you really want to delete it. An object reference to the removed node will be returned by the method. `childNode` is an object reference to the node to be removed.

Function replaceChild (newChild As ICMLDOMNode, oldChild As ICMLDOMNode) as IXMLDOMNode

The `removeChild` method is used to replace an existing node with a new node. The method will return an object reference to the old node.

`newChild` is an object reference to a new node. `oldChild` is an object reference to one of the current node's child nodes that will be replaced with `newChild`.

Function selectNodes (queryString As String) as IXMLDOMNodeList

The `selectNodes` method searches the current object and its children for elements that match the specified XSL pattern string, and returns them as an `IXMLDOMNodeList`. `queryString` is a `String` value containing an XSL Pattern query.

Function selectSingleNode (queryString As String) as IXMLDOMNode

The `selectSingleNode` method searches the current object and its children for the first element that matches the specified XSL pattern string, and returns it as an `IXMLDOMNode`. `queryString` is a `String` value containing an XSL Pattern query.

Function transformNode (stylesheet As IXMLDOMNode) as String

The `transformNode` method returns a `String` value containing a formatted XML document using the specified XSL style sheet. `stylesheet` is an object reference to an `IXMLDOMNode` object containing the root of the XSL style sheet.

Sub transformNodeToObject (stylesheet As IXMLDOMNode, outputObject)

The `transformNodeToObject` method creates a `DOMDocument` structure containing the formatted XML document using the specified XSL style sheet. `stylesheet` is an object reference to an `IXMLDOMNode` object containing the root of the XSL style sheet. `outputObject` is an object reference to a `DOMDocument` object which will contain the root object of the formatted document.

The DOMDocument Object

The `DOMDocument` object is the fundamental object in the Document Object Model. It represents a single XML document.

DOMDocument object properties

The `DOMDocument` object has all of the properties of the `XMLDOMNode` object (see Table 21-1), plus the additional ones listed in Table 21-3.

Table 21-3
Unique Properties of the `DOMDocument` Object

Property	*Description*
`async`	A `Boolean` value when `True` means that the `Load` method will return control to the caller before the load is complete. You must use the `readyState` property or the `onReadyStateChange` event to determine when the load process is finished.
`doctype`	An object reference to an `XMLDOMDocumentType` object (specified with the <!DOCTYPE> tag) containing the document type definition.
`documentElement`	An object reference to the root element of the document.
`implementation`	An object reference to the `XMLDOMImplementation` object, which contains information about the features that are supported in this implementation of DOM.

Property	*Description*
namespaces	An object reference to an XMLDOMSchemaCache object that contains the collection of namespaces used in this XML document.
ondataavailable	A Boolean value, when True means that the ondataavailable event is enabled.
onreadystatechange	A Boolean value, when True means that the onreadystatechange event is enabled.
ontransformnode	A Boolean value, when True means that the ontransformnode event is enabled.
parseError	An object reference to the XMLDOMParseError object, which contains information about the last parsing error.
preserveWhiteSpace	A Boolean, when True means that the parsing process will retain the blanks that are included in the XML source document.
readyState	A Long value describing the state of the document, when loading the document asynchronously (see the async property). A value of LOADING (1) means that the XML source document is being loaded; a value of LOADED (2) means that the document is loaded, but none of the objects are available for access; a value of INTERACTIVE (3) means that some objects are available for use; a value of COMPLETED (4) means that the document has been loaded and all objects are available for access. Note that a value of COMPLETED doesn't imply that the document loaded successfully.
resolveExternals	A Boolean value, when True means that external definitions such as namespaces, DTD external subsets, and external entity references should be resolved at parse time.
schemas	An object reference to an XMLSchemaCache object containing a list of schemas that should be used when loading an XML document.
url	A String value containing the URL that was used to load the XML document.
validateOnParse	A Boolean, when True means that the document's structure should be validated during parsing.

Enabled, but impossible: Simply setting the ondataavailable, onreadystatechange, and ontransformnode properties to True is not sufficient to enable the corresponding events. You must declare the Document object using the WithEvents keyword at the module level in your program in order to include code for the events.

DOMDocument object methods

The DOMDocument object has a number of unique methods that are used to create and access the information in an XML document.

Sub Abort ()

The Abort method will cancel an asynchronous download. This will return an error in the XMLDOMParseError object indicating that the download was aborted.

Function createAttribute (name As String) as IXMLDOMAttribute

The createAttribute method creates an empty attribute object with the specified name. Note that the newly created node must be added to another node using the appendChild method. name is a String value containing the name of the attribute.

Function createCDATASection (data As String) as IXMLDOMCDATASection

The createCDATASection method creates an empty IXMLDOMCDATASection object with the specified data. Note that the newly created node must be added to another node using the appendChild method. data is a String value that will be stored in the new object's nodeValue property.

Function createNode (type, name As String, namespaceURI As String) as IXMLDOMNode

The createNode method creates an empty node in the document. Note that the newly created node must be added to another node using the appendChild method. type is either a String value containing the type of the node or an Integer containing the numeric value corresponding to the type of the node (see Table 21-2) name is a String value containing the name of the node that will be stored in the nodeName property. namespaceURI is a String containing the URI of the namespace.

Function createProcessingInstruction (target As String, data As String) as IXMLDOMProcessingInstruction

The createProcessingInstruction method creates an empty IXMLDOMProcessingInstruction object with the specified data. Note that the newly created node must be added to another node using the appendChild method. target is a String value containing the name of the processing instruction. data is a String value that will be stored in the new object's nodeValue property.

Output version: Use this method to generate the <?xml version="1.0"?> element at the start of your XML document. Use `xml` as `target`, and `version="1.0"` as `data`.

Function createTextNode (data As String) as IXMLDOMTextNode

The `createTextNode` method creates an empty TextNode object with the specified data. Note that the newly created node must be added to another node using the `appendChild` method. `data` is a `String` value that will be stored in the new object's `nodeValue` property.

Function getElementsByTagName (tagName As String) as IXMLDOMNodeList

The `getElementsByTagName` method searches the current object and its children for the specified elements and returns them as an `IXMLDOMNodeList`. `tagName` is a `String` value containing the element name to be found. If you specify an asterisk (*), all elements will be returned.

Function getProperty(name As String)

The `getProperty` method returns a property value set by the `setProperty` method. `name` is a `String` value containing the name of the property. `SelectionLanguage` is a `String` value, which can be either Xpath or XSLPattern. It determines the type of query that the user specifies in the `selectNodes` or `selectSingleNode` methods.

Function load (url As String) as Boolean

The `load` method loads the specified XML document into the `DOMDocument`, parses it, and then creates the appropriate child objects to represent the XML document. If the load was successful, this method will return `True`. `url` is a `String` value containing a URL that specifies the location of the XML document to be loaded.

Function loadXML(xmlString As String) as Boolean

The `loadXML` method loads parses and then creates the appropriate child objects to represent the XML document specified in `xmlString`. If the load was successful, this method will return `True`. `xmlString` is a `String` value containing the XML statements to be loaded.

Function nodeFromId (idString As String) as IXMLDOMNode

The `nodeFromId` method returns the node, which has an ID attribute with the supplied value. The ID attribute is supposed to appear only once in an element, and the value of each ID attribute is supposed to be unique within an XML document. `idString` is a `String` value containing the ID value to be searched for.

Sub save (destination)

The `save` method is used to write the XML to the specified location. `Destination` is a `Variant` which can be a `String` value containing a file name or an ASP Response object, which can be used to send the document over the Internet in a VB IIS Application.

Sub setProperty(name As String, value)

The `setProperty` method allows you to set the value of the `SelectionLanguage` property described earlier in this chapter under "Function getProperty(name As String)." `name` is a `String` value containing the name of the property, whose value is to be changed with the `SelectionLanguage` property.. `value` is the value to be assigned to the property.

Sub validate()

The `validate` method verifies the currently loaded document against the currently loaded DTD or schema. Without a DTD or schema, the validate method will cause a run-time error.

DOMDocument object events

The `DOMDocument` object is unique in this object model in that it is the only object to have events. These events assist you in processing XML documents asynchronously.

Event ondataavailable ()

The `ondataavailable` event is called as soon as the first object containing data is available during an asynchronous load process. The `ondataavailable` event will be called as additional chunks of data become available. Note that the `ondataavailable` property must be set to `True`, and the `DOMDocument` object must be declared `WithEvents`, in order for the event to be fired.

Event onreadystatechange ()

The `onreadystatechange` event is called each time the `readyState` property changes value. Note that the `onreadystatechange` property must be set to `True`, and the `DOMDocument` object must be declared `WithEvents`, in order for the event to be fired.

Event Function ontransformnode (nodeCode, nodeData) As Boolean

The `ontransformnode` event is called before each node in the style sheet is applied to each node in the XML document. If you return `True`, the transformation process will

continue, while returning `False` will abort the transformation process. `nodeCode` is an object reference to the current node in the style sheet. `nodeData` is an object reference to the current node in the XML document.

Visual Basic, not: This event is not supported in Visual Basic due to the way it was implemented.

The XMLDOMAttribute object

The `XMLDOMAttribute` object holds information about a specific attribute in an element. Unlike the other node objects, the `XMLDOMAttribute` nodes are referenced through the `attributes` property.

XMLDOMAttribute object properties

Table 21-4 lists the unique properties associated with the `XMLDOMAttribute` object. All of the properties associated with the `XMLDOMNode` object (see Table 21-1) are also available in this object.

Table 21-4
Unique Properties of the XMLDOMNode object

Property	*Description*
`Name`	A `String` value containing the name of the attribute.
`Value`	A `String` containing the attribute's value.

XMLDOMAttribute object methods

The `XMLDOMAttribute` object has no unique methods. It inherits all of the methods found in the `XMLDOMNode` object.

The XMLDOMCDATASection Object

The `XMLDOMComment` object holds information about a CDATA section in your XML document.

XMLDOMCDATASection object properties

Table 21-5 lists the unique properties associated with the `XMLDOMCDATASection` object. All of the properties listed in the `XMLDOMNode` object (see Table 21-1) are also available for this object.

Table 21-5
Unique Properties of the XMLDOMCDATASection Object

Property	*Description*
`Data`	A `String` value containing the characters from the CDATA section of the XML document.
`Length`	A `Long` value containing the number of characters in `data`.

XMLDOMCDATASection object methods

The `XMLDOMCDATASection` object has two unique methods that are used to access the information from a CDATA section in your XML document. All of the methods found in the `XMLDOMNode` object are available for this object as well.

Function splitText (offset As Long) as IXMLDOMText

The `splitText` method splits the node into two nodes at the specified offset from the beginning of the text and automatically inserts the new node into the document hierarchy immediately following the current node. `offset` is a `Long` value containing the location where the split will take place. Specifying an offset value of zero will move all of the text in the current node to the new node.

Function substringData (offset As Long, count As Long) as String

The `substringData` method extracts a block of text from the node. `offset` is a `Long` value containing the location where the extraction will begin. The first character in the string has an offset of zero. `count` is a `Long` containing the number of characters to be extracted.

The XMLDOMComment Object

The `XMLDOMComment` object holds information about a comment in your XML document.

XMLDOMComment object properties

Table 21-6 lists the properties associated with the `XMLDOMComment` object. It also inherits the properties from the `XMLDOMNode` object (see Table 21-1).

Table 21-6
Unique Properties of the XMLDOMComment Object

Property	*Description*
`Data`	A `String` value containing the characters in the Comment section of the XML document.
`Length`	A `Long` value containing the number of characters in `data`.

XMLDOMComment object methods

The `XMLDOMComment` object has a method that you can use to extract information from a Comment section in your XML document. The methods available in the `XMLDOMNode` object are also available in this object.

Function substringData (offset As Long, count As Long) as String

The `substringData` method extracts a block of text from the node. `offset` is a `Long` value containing the location where the extraction will begin. The first character in the string has an offset of zero. `count` is a `Long` containing the number of characters to be extracted.

The XMLDOMDocumentType Object

The `XMLDOMDocumentType` object holds information about a document type declaration.

XMLDOMDocumentType object properties

Table 21-7 lists the unique properties associated with the `XMLDOMDocumentType` object.

Table 21-7
Unique Properties of the XMLDOMNode Object

Property	*Description*
`Entities`	An object reference to an `XMLDOMNameNodeMap` containing the collection of entities used in the document type declaration.
`Name`	A `String` value containing the name of the document type.
`Notations`	An object reference to an `XMLDOMNameNodeMap` containing the collection of `XMLDOMNotation` objects.

XMLDOMDocumentType object methods

The `XMLDOMDocumentType` object has no unique methods. It inherits all of the methods found in the `XMLDOMNode` object.

The XMLDOMElement Object

The `XMLDOMElement` object holds information about an entity from your XML document.

XMLDOMElement object properties

The unique property of the `XMLDOMElement` object is `tagName`, which is a `String` value containing the name of the tag that is used to identify the element. The `XMLDOMElement` object also inherits the properties from the `XMLDOMNode` object (see Table 21-1).

XMLDOMElement object methods

The `XMLDOMElement` object provides several methods that make it easy to access the information from the attributes associated with this object. Note that the methods available in the `XMLDOMNode` object (see Table 21-1) are also available in this object.

Function getAttribute (name as String) as Variant

The `getAttribute` method returns a `String` containing the value of the attribute. An empty string means that the attribute doesn't have a specified or default value. `name` is a `String` containing the name of the attribute.

Function getAttributeNode (name as String) as IXMLDOMAttribute

The `getAttributeNode` method returns a reference to an `XMLDOMAttribute` object containing the specified attribute. `name` is a `String` value containing the name of the attribute.

Sub normalize()

The `normalize` method combines all of the text nodes below this object into normal form, where each text node is separated by an element, a comment, a processing instruction, a CDATA section, or an entity reference. You can do the same thing by saving the document to a disk file, deleting all of the objects below the `DOMDocument` object, and loading the document back from the disk file.

Sub setAttribute (name as String, value)

The `setAttribute` method assigns a new value to an attribute. If the attribute doesn't exist, it will automatically be created for you. `name` is a `String` value containing the name of the attribute. `value` is a `Variant` value containing the value to be assigned to the attribute.

Function setAttributeNode (DOMAttribute as IXMLDOMAttribute) as IXMLDOMAttribute

The `setAttributeNode` method is used to create or replace an attribute node in the document hierarchy. It will return `Null` if the attribute didn't exist, or it will return an object pointer to the old attribute's node. `DOMAttribute` is an object reference to an `XMLDOMAttribute` object that contains the new attribute.

The XMLDOMEntity Object

The `XMLDOMEntity` object holds information about an entity from your XML document.

XMLDOMEntity object properties

Table 21-8 lists the unique properties of the `XMLDOMEntity` object. This object also inherits all of the properties from the `XMLDOMNode` object (see Table 21-1).

Table 21-8
Unique Properties of the XMLDOMEntity Object

Property	*Description*
`notationName`	A `String` value containing the name of the notation, if the entity is unparsed or an empty string once the entity has been parsed.
`publicId`	A `String` containing the public identifier associated with the entity.
`systemId`	A `String` containing the system identifier associated with the entity. If the system identifier isn't specified, this property will contain the empty string.

XMLDOMEntity object methods

The `XMLDOMEntity` object has no unique methods. It inherits all of the methods found in the `XMLDOMNode` object.

The XMLDOMEntityReference Object

The `XMLDOMEntityReference` object holds information about an entity from your XML document. This object is created based on the information from the `XMLDOMEntity` object. Due to the nature of the XML parser included in the object library, external entities may not be parsed and expanded in their own objects until they are needed, which means that two different objects are needed to hold the information.

No difference: The `XMLDOMEntityReference` object doesn't have any unique properties, methods, or events, when compared to the `XMLDOMNode` object. All of the standard properties and methods of the `XMLDOMNode` object are available in this object.

The XMLDOMNotation Object

The `XMLDOMNotation` object holds information about a notation that was declared in the data type declaration or schema section of your XML document.

XMLDOMNotation object properties

Table 21-9 lists the unique properties of the `XMLDOMNotation` object. This object also inherits all of the properties from the `XMLDOMNode` object.

Table 21-9
Unique Properties of the XMLDOMNotation Object

Property	*Description*
`publicId`	A `String` value containing the public identifier associated with the notation.
`systemId`	A `String` containing the system identifier associated with the notation. If the system identifier isn't specified, this property will contain the empty string.

XMLDOMNotation object methods

The `XMLDOMNotation` object has no unique methods. It inherits all of the methods found in the `XMLDOMNode` object (see Table 21-1).

The XMLDOMProcessingInstruction Object

The `XMLDOMProcessingInstruction` object contains processing directives, such as the `<?xml version=1.0?>`.

XMLDOMProcessingInstruction object properties

Table 21-10 lists the unique properties of the `XMLDOMProcessingInstruction` object. This object also inherits all of the properties from the `XMLDOMNode` object (see Table 21-1).

XMLDOMProcessingInstruction object methods

The `XMLDOMProcessingInstruction` object has no unique methods. It inherits all of the methods found in the `XMLDOMNode` object.

Table 21-10
Unique Properties of the XMLDOMProcessingInstruction Object

Property	*Description*
`data`	A `String` value containing the data associated with the processing instruction. In other words, this would be the `version="1.0"` part of the processing instruction. This value is identical to that in the `nodeValue` property.
`target`	A `String` containing the application that contains the processing instruction. For example, this would be `xml` from the `<?xml version="1.0"?>` processing instruction.

The XMLDOMText Object

The `XMLDOMText` object holds information about the text value associated with an `XMLDOMElement` or `XMLDOMAttribute` node.

XMLDOMText object properties

Table 21-11 lists the unique properties associated with the `XMLDOMText` object. All of the properties listed in the `XMLDOMNode` object are also available for this object.

Table 21-11
Unique Properties of the XMLDOMText Object

Property	*Description*
`data`	A `String` value containing the text characters that make up the value of an element or an attribute.
`length`	A `Long` value containing the number of characters in `data`.

XMLDOMText object methods

The `XMLDOMText` object includes a rich set of methods to manipulate the data stored in the node. It also includes all of the methods found in the `XMLDOMNode` object.

Sub appendData (data as String)

The `appendData` method adds the specified string value to the end of the existing data already in the node. `data` is a `String` value containing the new information to be added to the node's value.

Sub deleteData (offset as Long, count as Long)

The `deleteData` method removes the specified number characters from the node starting with the specified location. `offset` is a `Long` value containing the location where the characters will be deleted. The first character in the string has an offset of zero. `count` is a `Long` containing the number of characters to be deleted.

Sub insertData (offset as Long, data as String)

The `insertData` method inserts the specified data into the data already starting with the specified location. `offset` is a `Long` containing the location where the characters will be inserted. The first character in the string has an offset of zero. `data` is a `String` containing the characters to be added.

Sub replaceData (offset as Long, count as Long, data as String)

The `replaceData` method deletes the specified number of characters starting at the specified location, then inserts the specified data starting at the same location. `offset` is a `Long` containing the location where the characters will be replaced. The first character in the string has an offset of zero. `count` is a `Long` containing the number of characters to be deleted. `data` is a `String` containing the characters to be added.

Function splitText (offset as Long) as IXMLDOMText

The `splitText` method splits the node into two nodes at the specified offset from the beginning of the text, and automatically inserts the new node into the document hierarchy immediately following the current node. `offset` is a `Long` containing the location where the split will take place. Specifying an offset value of zero will move all of the text in the current node to the new node.

Function substringData (offset as Long, count as Long) as String

The `substringData` method extracts a block of text from the node. `offset` is a `Long` value containing the location where the extraction will begin. The first character in the string has an offset of zero. `count` is a `Long` containing the number of characters to be extracted.

The XMLDOMParseError Object

The XMLDOMParseError object contains information about the first error encountered while parsing your XML document.

XMLDOMParseError object properties

Table 21-12 lists the properties of the XMLDOMParseError object.

Table 21-12
Properties of the XMLDOMParseError object

Property	*Description*
ErrorCode	A Long value containing the error code.
Filepos	A Long containing the absolute position in the file where the error occurred.
Line	A Long containing the line number where the error occurred.
Linepos	A Long containing the position in the line where the error occurred.
Reason	A String value containing a text description of the error.
SrcText	A String containing the line with the error.
url	A String containing the URL of the document with the error.

XMLDOMParseError bject methods

The XMLDOMParseError object doesn't have any methods.

The XMLHttpRequest Object

The XMLHttpRequest object allows you to transfer XML documents using HTTP. This is primarily a client-side tool that can be used in a JavaScript or VBScript Web page to transmit a request to a server and process its response. The properties and objects that return information about the result are only valid if the send method has completed successfully.

XMLHttpRequest object properties

Table 21-13 lists the properties of the `XMLHttpRequest` object.

Table 21-13
Properties of the XMLHttpRequest Object

Property	*Description*
`onreadystatechange`	An object reference to an event handler in a scripting language.
`readyState`	A `Long` value describing the state of the transport, when loading the document asynchronously (see the `async` property). A value of `UNINITIALIZED` (0) means that the object has been created but nothing has been transferred. A value of `LOADING` (1) means that the source document is being loaded; a value of `LOADED` (2) means that the document is loaded, but none of the objects in the document hierarchy are available for access; a value of `INTERACTIVE` (3) means that some objects are available for use; a value of `COMPLETED` (4) means that the document has been loaded and all objects are available for access. Note that a value of `COMPLETED` doesn't imply that the document loaded successfully.
`responseBody`	A `Variant` value containing the response to the HTTP request.
`responseStream`	An object reference to a `Stream` object containing the raw data from the response to the HTTP request.
`responseText`	A `String` value containing the response to the HTTP request.
`responseXML`	An object reference to a `DOMDocument` object containing the parsed XML document that was received in response to the HTTP request.
`status`	A `Long` containing the HTTP status code returned by the HTTP server.
`statusText`	A `String` containing the HTTP line status.

XMLHttpRequest object methods

The `XMLHttpRequest` object includes methods for creating and sending an HTTP request to a Web server.

Sub abort()

The `abort` method terminates an active HTTP request.

Function getAllResponseHeaders () as String

The `getAllResponseHeaders` method returns the header information from the HTTP result as a single string. Each header is separated by a carriage return/line feed pair (`vbCrLf`).

Function getResponseHeader (bstrHeader as String) as String

The `getResponseHeader` method returns the specified header from the HTTP response where `bstrHeader` is a `String` value containing the particular header you wish to retrieve.

Sub open(bstrMethod as String, bstrUrl As String, [varAsync], [bstrUser], [bstrPassword])

The `open` method initializes an HTTP request. After the request is initialized, the `send` method must be used to transfer the document and wait for the response. `bstrMethod` is a `String` value that identifies the HTTP transfer method. This is usually one of the following: `GET`, `POST`, `PUT`, or `PROPFIND`. `bstrUrl` is a `String` containing the URL that will be used to process the request. `varAsync` is a `Boolean` value when `True` means that the call is asynchronous. If this value isn't specified, it will default to `True`. `bstrUser` is a `String` containing the userid to log onto the Web server. If this parameter is missing and the Web server requires a userId and password to log on, the user will be prompted for this information. `bstrPassword` is a `String` containing the password associated with `bstrUserid`.

Sub send ([varBody])

The `send` method transmits the request to the remote host and optionally waits for the host's response if the user specified `False` for `varAsync` on the `open` method. `varBody` is a `Variant` value containing the document to be sent. If this parameter isn't specified, then the current document is transmitted.

Sub setRequestHeader(bstrHeader as String, bstrValue as String)

The `setRequestHeader` method sets the value of the various header fields before the document request is sent. `bstrHeader` is a `String` value containing the header to be set. Note that the trailing colon (:) on the header should not be specified. `bstrValue` is a `String` containing the value to be assigned to the header.

Thoughts on the XML Object Model

As I said in the previous chapter, XML is far more complex than I have room to cover. This is really obvious when you look at the object model. However, much of the complexity isn't necessary for most programmers. As with most things in life, the eighty-twenty rule applies: eighty percent of the time, only twenty percent of the capabilities are needed or used.

In this case, the object model is designed to cover all sorts of situations that might not normally be encountered in most applications. In the next chapter, I'm going to use this object model in a simple example that shows you how you can generate an XML request and satisfy it with a Web server.

Summary

In this chapter you learned:

- about the XML Document Object Model
- how an XML document is mapped into the Document Object Model.

✦ ✦ ✦