Chapter 8

XML

Chapter 8
XML

What Is XML?

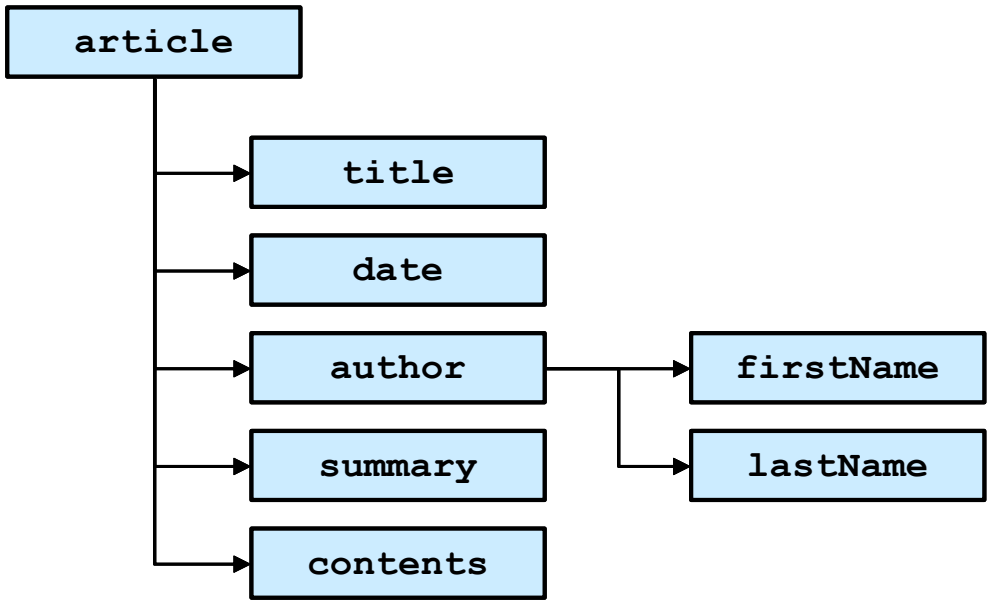
XML is a text-based way to describe structured data. Unlike HTML, which uses various markup tags to describe how data should be displayed or rendered in a Web browser, XML is simply data.

```
1   <?xml version = "1.0"?>
3   <!-- letter.xml          Business letter formatted with XML   -->
5
6   <letter>
7     <contact type = "from">
8       <name>Jane Doe</name>
9       <address1>Box 12345</address1>
10      <address2>15 Any Ave.</address2>
11      <city>Othertown</city>
12      <state>Otherstate</state>
13      <zip>67890</zip>
14      <phone>555-4321</phone>
15      <flag gender = "F" />
16    </contact>
17
18    <contact type = "to">
19      <name>John Doe</name>
20      <address1>123 Main St.</address1>
21      <address2></address2>
22      <city>Anytown</city>
23      <state>Anystate</state>
24      <zip>12345</zip>
25      <phone>555-1234</phone>
26      <flag gender = "M" />
27    </contact>
```

```
29     <salutation>Dear Sir:</salutation>
30
31       <paragraph>It is our privilege to inform you about our new
32       database managed with <technology>XML</technology>.
This
33       new system allows you to reduce the load on
34       your inventory list server by having the client machine
35       perform the work of sorting and filtering the data.
36       </paragraph>
37
38       <paragraph>Please visit our Web site for availability
39       and pricing.
40       </paragraph>
41     <closing>Sincerely</closing>
42     <signature>Ms. Doe</signature>
43   </letter>
```

Document Object Model (DOM)

Store document data as tree structure (DOM tree). Each component is a node in the tree (root node. and Parent, sibling, descendent nodes)



```
1   <?xml version = "1.0"?>
3   <!-- article.xml      -->

6   <article>
8      <title>Simple XML</title>
10     <date>December 6, 2009</date>
12     <author>
13        <firstName>Mohammed </firstName>
14        <lastName>El-dosuky </lastName>
15     </author>
17     <summary>XML easy</summary>
19     <content>In this chapter, we present a wide variety of
examples that use XML </content>
20   </article>
```

```
1   // XmlReaderTest.cs:        Reading an XML document.
4   using System;
5   using System.Windows.Forms;
6   using System.Xml;
8   public class XmlReaderTest : System.Windows.Forms.Form
9   {
10     private System.Windows.Forms.TextBox outputTextBox;
12
13     public XmlReaderTest()
14     {
15        InitializeComponent();
16
18        XmlDocument document = new XmlDocument(); //
19        document.Load( "article.xml" );          // Parse it
22     XmlNodeReader reader = new XmlNodeReader( document );
// create XmlNodeReader
23
25        this.Show();   // show form before outputTextBox
26
28        int depth = -1;    // tree depth is -1, no indentation
```

```
31        while ( reader.Read() )    // display each node's content
32        {
33          switch ( reader.NodeType )
34          {
36            case XmlNodeType.Element: //  display name
38              // increase tab depth
39              depth++;              TabOutput( depth );
41              outputTextBox.Text += "<" + reader.Name + ">" +
"\r\n";
44              // if empty element, decrease depth
45              if ( reader.IsEmptyElement )   depth--;
48              break;
51            case XmlNodeType.Comment: // if Comment, display
52              TabOutput( depth );
53          outputTextBox.Text +=  "<!--" + reader.Value + "-->\r\n";
55              break;
56
58            case XmlNodeType.Text:  // if Text, display it
59              TabOutput( depth );
60              outputTextBox.Text += "\t" + reader.Value + "\r\n";
62              break;
63
65            case XmlNodeType.XmlDeclaration:  // display it
66              TabOutput( depth );
67    outputTextBox.Text += "<?" + reader.Name
68               +""+ reader.Value + " ?>\r\n";
69              break;
70
72            case XmlNodeType.EndElement:  // decrement depth
73              TabOutput( depth );
74              outputTextBox.Text += "</" + reader.Name + ">\r\n";
76              depth--;
77              break;
78          } // end switch statement
79        } // end while loop
80      } // End XmlReaderTest constructor
81
```

```
83     private void TabOutput( int number )
84     {
85        for ( int i = 0; i < number; i++ )
86              outputTextBox.Text += "\t";
87     }
96   } // end XmlReaderTest
```

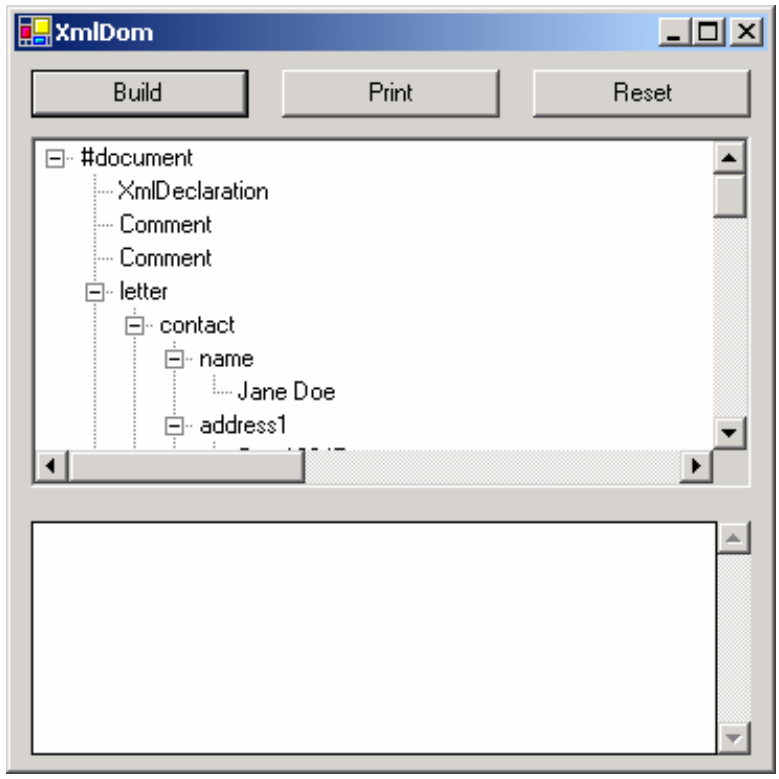
```
1    // XmlDom.cs        :        Demonstrates DOM tree
manipulation.
4    using System;
5    using System.Windows.Forms;
6    using System.Xml;
7    using System.IO;
8    using System.CodeDom.Compiler;  // contains
TempFileCollection
10
11   public class XmlDom : System.Windows.Forms.Form
12   {
13      private System.Windows.Forms.Button buildButton;
14      private System.Windows.Forms.Button printButton;
15      private System.Windows.Forms.TreeView xmlTreeView;
16      private System.Windows.Forms.TextBox consoleTextBox;
17      private System.Windows.Forms.Button resetButton;
19
20      private XmlDocument source; // XML document
23      private XmlDocument copy;    // copy of source's
25      private TreeNode tree; // TreeNode reference
27      public XmlDom()
28      {
29         InitializeComponent();
30
32         source = new XmlDocument();  // create XmlDocument
and load letter.xml
33         source.Load( "letter.xml" );
35
```

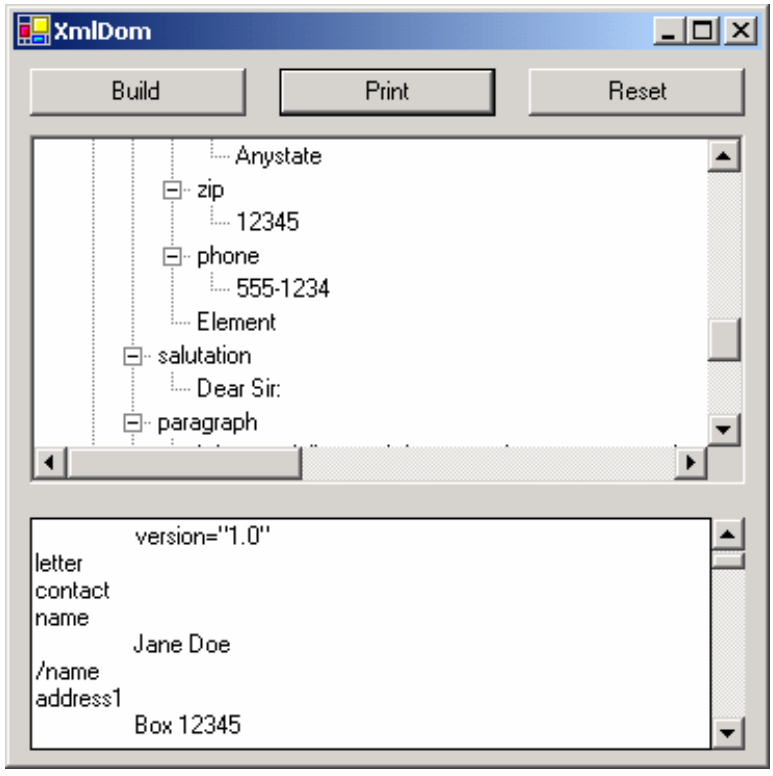
```
36        copy = null;
37       tree = null;    // set references to null
38     }

47     private void buildButton_Click( object sender,
System.EventArgs e )
49     {
51       if ( copy != null ) return;         // if copy has been built
52
55       copy = new XmlDocument();  tree = new TreeNode();
56      //create XmlDocument, TreeNode
57
60       tree.Text = source.Name;      // assigns the root
61       xmlTreeView.Nodes.Add( tree );
62
64       BuildTree( source, copy, tree ); // build node and tree
65
66       printButton.Enabled =  resetButton.Enabled = true;
68     }
71     private void printButton_Click( object sender,
System.EventArgs e )
73     {
74        if ( copy == null )    return;
77
79      TempFileCollection file = new TempFileCollection();
//              temp file
80
82       file.AddExtension( "xml", false ); // deleted at termination
83       string[] filename = new string[ 1 ];
84       file.CopyTo( filename, 0 );
85
87       XmlTextWriter writer=new XmlTextWriter( filename[0],
System.Text.Encoding.UTF8);
89       copy.WriteTo( writer ); // write XML data to disk
90       writer.Close();
91
93       XmlTextReader reader=new XmlTextReader(filename[0]);
```

```
94      //parse,load temp document
96       while( reader.Read() )
97       {
98         if ( reader.NodeType == XmlNodeType.EndElement )
consoleTextBox.Text += "/";
100
101       if ( reader.Name != String.Empty )
consoleTextBox.Text += reader.Name + "\r\n";
103
104       if ( reader.Value != String.Empty )  consoleTextBox.Text
+= "\t" +reader.Value +"\r\n";
107      }
109      reader.Close();
110    }
113    private void resetButton_Click( object sender,
System.EventArgs e )
115    {
117      if ( tree != null )     xmlTreeView.Nodes.Remove( tree );
// remove TreeView nodes
119
120      xmlTreeView.Refresh(); // force TreeView update
121
122      copy = null;
123      tree = null;
124     // delete XmlDocument and tree
126      consoleTextBox.Text = "";
128      printButton.Enabled = resetButton.Enabled = false;
131    }
134    void BuildTree(XmlNode xmlSourceNode, XmlNode
document, TreeNode treeNode )
136    {
138      XmlNodeReader nodeReader = new
XmlNodeReader(xmlSourceNode );
142      XmlNode currentNode = null;
145      TreeNode newNode = new TreeNode();
148      XmlNodeType modifiedNodeType; // references modified
node type for CreateNode
```

```
150     while ( nodeReader.Read() )
151     {
153       modifiedNodeType = nodeReader.NodeType;
154
156       if ( modifiedNodeType == XmlNodeType.EndElement )
157         modifiedNodeType = XmlNodeType.Element;
158
160       currentNode = copy.CreateNode( modifiedNodeType,
161             nodeReader.Name, nodeReader.NamespaceURI );
162
164       switch ( nodeReader.NodeType ) // build tree
165       {
167         case XmlNodeType.Text:
168           newNode.Text = nodeReader.Value;
169           treeNode.Nodes.Add( newNode );
171           // append Text node value to currentNode data
172           ( ( XmlText ) currentNode ).AppendData(
173                       nodeReader.Value );
174           document.AppendChild( currentNode );
175           break;
176
178         case XmlNodeType.EndElement: // if EndElement,
179           document = document.ParentNode;
180           treeNode = treeNode.Parent;
181           break;
183
184         case XmlNodeType.Element:
185          // if new element, add name and traverse tree
187           if ( ! nodeReader.IsEmptyElement )
188 // determine if element contains content
189           {
190             newNode.Text = nodeReader.Name;
191             treeNode.Nodes.Add( newNode );
194             treeNode = newNode;
196             document.AppendChild( currentNode );
197             document = document.LastChild;
198           }
```

```
199             else // do not traverse empty elements
200             {
201               // assign NodeType string to newNode
202           newNode.Text =  nodeReader.NodeType.ToString();
204
205               treeNode.Nodes.Add( newNode );
206               document.AppendChild( currentNode );
207             }
209             break;
210
212           default:        // all other types, display node type
213             newNode.Text = nodeReader.NodeType.ToString();
214             treeNode.Nodes.Add( newNode );
215             document.AppendChild( currentNode );
216             break;
217         }
219         newNode = new TreeNode();
220       } // end while
223       xmlTreeView.ExpandAll();  xmlTreeView.Refresh();
224           // update the TreeView control
225
226     } // end BuildTree
227   } // end XmlDom
```


XmlDom
Build
Print
Reset
#document
XmlDeclaration
Comment
Comment
letter
contact
name
Jane Doe
address1


XmlDom
Build
Print
Reset
Anystate
zip
12345
phone
555-1234
Element
salutation
Dear Sir:
paragraph
version="1.0"
letter
contact
name
Jane Doe
/name
address1
Box 12345

```
1   // PathNavigator.cs
4   using System;
5   using System.Windows.Forms;
6   using System.Xml.XPath; // contains XPathNavigator
8   public class PathNavigator : System.Windows.Forms.Form
9   {
10     private System.Windows.Forms.Button firstChildButton;
11     private System.Windows.Forms.Button parentButton;
12     private System.Windows.Forms.Button nextButton;
13     private System.Windows.Forms.Button previousButton;
14     private System.Windows.Forms.Button selectButton;
15     private System.Windows.Forms.TreeView pathTreeViewer;
16     private System.Windows.Forms.ComboBox
selectComboBox;
18     private System.Windows.Forms.TextBox selectTreeViewer;
19     private System.Windows.Forms.GroupBox navigateBox;
20     private System.Windows.Forms.GroupBox locateBox;
23     private XPathNavigator xpath; // navigator to traverse
document
26     private XPathDocument document;
29     private TreeNode tree;
31     public PathNavigator()
32     {
33        InitializeComponent();
34
36        document = new XPathDocument( "sports.xml" );
39        xpath = document.CreateNavigator();
42        tree = new TreeNode();
43
44        tree.Text = xpath.NodeType.ToString(); // #root
45        pathTreeViewer.Nodes.Add( tree );      // add tree
47
48        pathTreeViewer.ExpandAll();  pathTreeViewer.Refresh();
49          // update TreeView control
50        pathTreeViewer.SelectedNode = tree;    // highlight root
51     }
60     private void firstChildButton_Click( object sender,
System.EventArgs e )
```

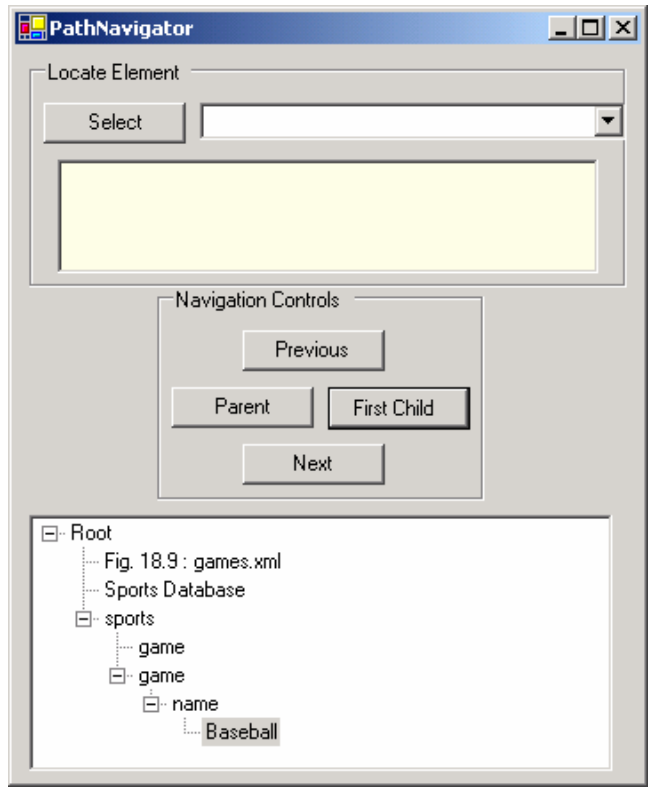
```
62     {
66        if ( xpath.MoveToFirstChild() ) // move to first child
67        {
68           TreeNode newTreeNode = new TreeNode(); // create
new node
72           DetermineType( newTreeNode, xpath ); // set node Text
to navigator's name or value
73
75           tree.Nodes.Add( newTreeNode ); // add node to
TreeNode node list
76           tree = newTreeNode; // assign tree newTreeNode
77
79           pathTreeViewer.ExpandAll();  pathTreeViewer.Refresh();
// update TreeView control
81           pathTreeViewer.SelectedNode = tree;
82        }
83        else // node has no children
84           MessageBox.Show( "Current Node has no children.", "",
MessageBoxButtons.OK,
86              MessageBoxIcon.Information );
87     }
90     private void parentButton_Click( object sender,
System.EventArgs e )
92     {
93        if ( xpath.MoveToParent() )
95        {
96           tree = tree.Parent;
99           int count = tree.GetNodeCount( false );
100    // get # of child nodes, not including sub trees
101
102          tree.Nodes.Clear();    // remove all children
105          pathTreeViewer.ExpandAll(); pathTreeViewer.Refresh();
106             // update TreeView control
107          pathTreeViewer.SelectedNode = tree;
108       }
109       else // if node has no parent (root node)
```

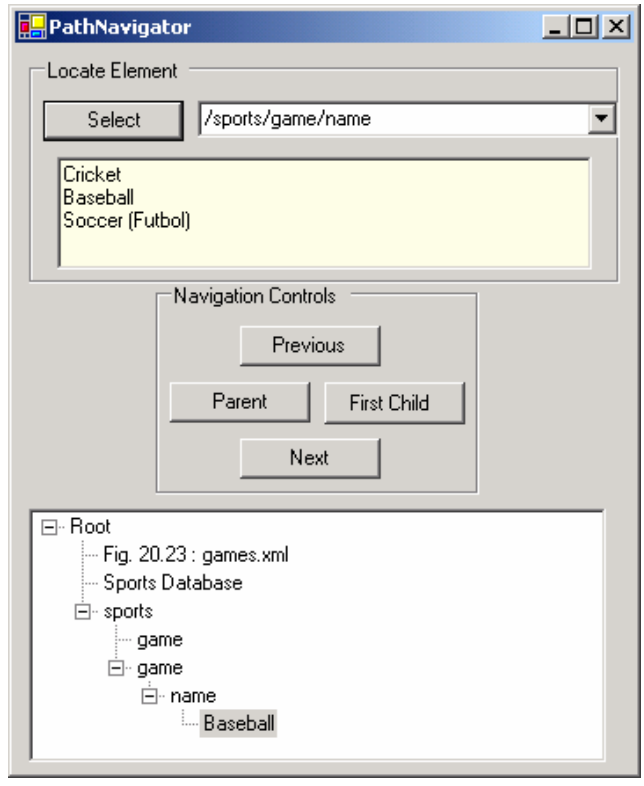
```
110         MessageBox.Show( "Current node has no parent.", "",
111            MessageBoxButtons.OK,
112            MessageBoxIcon.Information );
113     }
116     private void nextButton_Click( object sender,
System.EventArgs e )
118     {
122       if ( xpath.MoveToNext() )
123       {
124         TreeNode newTreeNode = tree.Parent;
125         TreeNode newNode = new TreeNode();
126
127         DetermineType( newNode, xpath );
128         newTreeNode.Nodes.Add( newNode );
131         tree = newNode;       // set current position for display
134         pathTreeViewer.ExpandAll(); pathTreeViewer.Refresh();
135            // update TreeView control
136         pathTreeViewer.SelectedNode = tree;
137       }
138       else // node has no additional siblings
139         MessageBox.Show( "Current node is last sibling.", "",
140            MessageBoxButtons.OK,
141            MessageBoxIcon.Information );
142     }
144     private void previousButton_Click( object sender,
System.EventArgs e )
147     {
151       if ( xpath.MoveToPrevious() )
152       {
153         TreeNode parentTreeNode = tree.Parent;
156         parentTreeNode.Nodes.Remove( tree ); // delete node
158
159         tree = parentTreeNode.LastNode; // to previous node
162         pathTreeViewer.ExpandAll(); pathTreeViewer.Refresh();
163            // update TreeView control
164         pathTreeViewer.SelectedNode = tree;
165       }
```

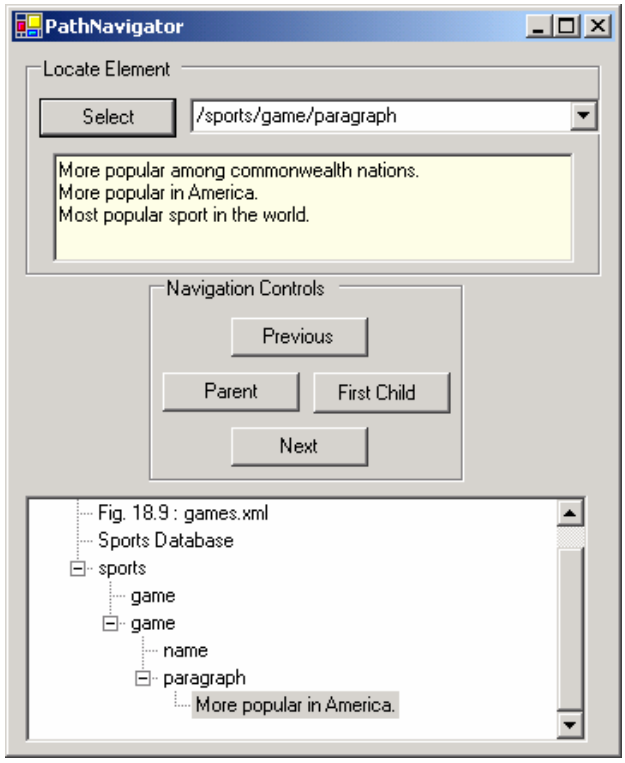
```
166       else // if current node has no previous siblings
167          MessageBox.Show( "Current node is first sibling.", "",
168             MessageBoxButtons.OK,
169             MessageBoxIcon.Information );
170    }
173    private void selectButton_Click( object sender,
System.EventArgs e )
175    {
179       try          // get specified node from ComboBox
180       {
181          XPathNodeIterator iterator = xpath.Select(
selectComboBox.Text );
182          DisplayIterator( iterator ); // print selection
183       }
186       catch ( System.ArgumentException ex)
187       {
188          MessageBox.Show(ex.Message, "Error",
189             MessageBoxButtons.OK,
190             MessageBoxIcon.Error );
191       }
192    }
195    private void DisplayIterator( XPathNodeIterator iterator)
196    {
197       selectTreeViewer.Text = "";
200       while (iterator.MoveNext()) selectTreeViewer.Text +=
iterator.Current.Value.Trim() + "\r\n";
204    }
208    private void DetermineType( TreeNode node,
209                                XPathNavigator xPath )
210    {
212       switch ( xPath.NodeType ) // determine NodeType
213       {
215          case XPathNodeType.Element: // if, get its name
218           node.Text = xPath.Name.Trim();  // remove whitespace
219             break;
222          default:          // obtain node values
225             node.Text = xPath.Value.Trim();
```

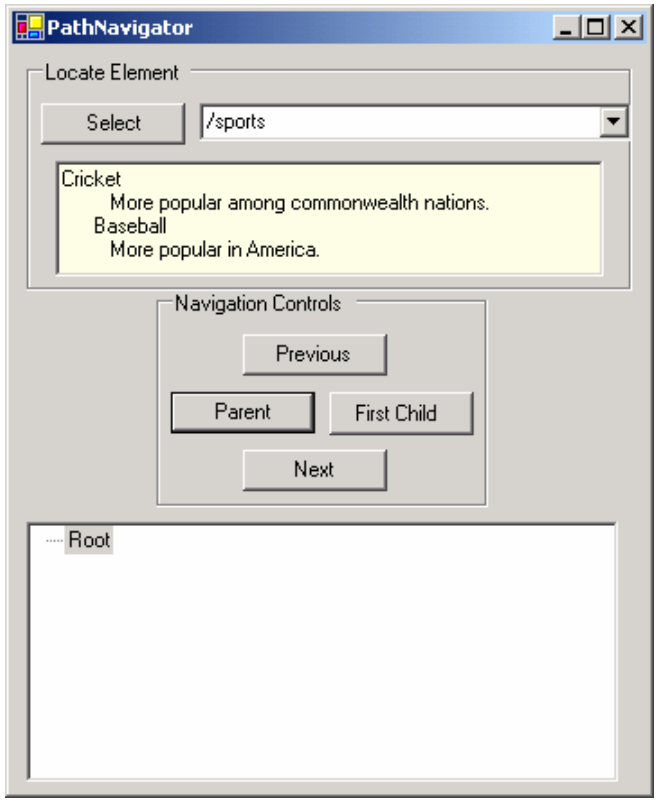
```
226             break;
228         } // end switch
229     } // end DetermineType
231 } // end PathNavigator
```

```
1   <?xml version = "1.0"?>
3   <!-- games.xml   Sports Database  -->
6   <sports>
8       <game id = "783">
9               <name>Cricket</name>
11              <paragraph> More popular among commonwealth
nations.</paragraph>
14      </game>
16      <game id = "239">
17              <name>Baseball</name>
19              <paragraph> More popular in America.</paragraph>
22      </game>
24      <game id = "418">
25              <name>Soccer(Futbol)</name>
26              <paragraph>Most popular sport in the
world</paragraph>
27      </game>
28  </sports>
```


PathNavigator
Locate Element
Select
Navigation Controls
Previous
Parent
First Child
Next
Root
Fig. 18.9 : games.xml
Sports Database
sports
game
game
name
Baseball


PathNavigator
Locate Element
Select
/sports/game/name
Cricket
Baseball
Soccer (Futbol)
Navigation Controls
Previous
Parent
First Child
Next
Root
Fig. 20.23 : games.xml
Sports Database
sports
game
game
name
Baseball


PathNavigator
Locate Element
Select
/sports/game/paragraph
More popular among commonwealth nations.
More popular in America.
Most popular sport in the world.
Navigation Controls
Previous
Parent
First Child
Next
Fig. 18.9 : games.xml
Sports Database
sports
game
game
name
paragraph
More popular in America.


PathNavigator
Locate Element
Select
/sports
Cricket
More popular among commonwealth nations.
Baseball
More popular in America.
Navigation Controls
Previous
Parent
First Child
Next
Root