Introduction to XML

XML was designed to describe data and to focus on what data is.

HTML was designed to display data and to focus on how data looks.

What is XML?

- XML stands for E**X**tensible **M**arkup **L**anguage
- XML is a **markup language** much like HTML
- XML was designed to **describe data**
- XML tags are not predefined. You must **define your own tags**
- XML uses a **Document Type Definition** (DTD) or an **XML Schema** to describe the data
- XML with a DTD or XML Schema is designed to be **self-descriptive**

The main difference between XML and HTML

XML was designed to carry data.

XML is not a replacement for HTML.
XML and HTML were designed with different goals:

XML was designed to describe data and to focus on what data is.
HTML was designed to display data and to focus on how data looks.

HTML is about displaying information, while XML is about describing information.

XML does not DO anything

XML was not designed to DO anything.

Maybe it is a little hard to understand, but XML does not DO anything. XML was created to structure, store and to send information.

The following example is a note to Tove from Jani, stored as XML:

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

The note has a header and a message body. It also has sender and receiver information. But still, this XML document does not DO anything. It is just pure information wrapped in XML tags. Someone must write a piece of software to send, receive or display it.

XML is free and extensible

XML tags are not predefined. You must "invent" your own tags.

The tags used to mark up HTML documents and the structure of HTML documents are predefined. The author of HTML documents can only use tags that are defined in the HTML standard (like <p>, <h1>, etc.).

XML allows the author to define his own tags and his own document structure.

The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.

XML is a complement to HTML

XML is not a replacement for HTML.

It is important to understand that XML is not a replacement for HTML. In future Web development it is most likely that XML will be used to describe the data, while HTML will be used to format and display the same data.

My best description of XML is this: **XML is a cross-platform, software and hardware independent tool for transmitting information.**

XML in future Web development

XML is going to be everywhere.

We have been participating in XML development since its creation. It has been amazing to see how quickly the XML standard has been developed and how quickly a large number of software vendors have adopted the standard.

We strongly believe that XML will be as important to the future of the Web as HTML has been to the foundation of the Web and that XML will be the most common tool for all data manipulation and data transmission.

How can XML be Used?

XML can Separate Data from HTML

With XML, your data is stored outside your HTML.

When HTML is used to display data, the data is stored inside your HTML. With XML, data can be stored in separate XML files. This way you can concentrate on using HTML for data layout and display, and be sure that changes in the underlying data will not require any changes to your HTML.

XML data can also be stored inside HTML pages as "Data Islands". You can still concentrate on using HTML only for formatting and displaying the data.

XML is used to Exchange Data

With XML, data can be exchanged between incompatible systems.

In the real world, computer systems and databases contain data in incompatible formats. One of the most time-consuming challenges for developers has been to exchange data between such systems over the Internet.

Converting the data to XML can greatly reduce this complexity and create data that can be read by many different types of applications.

XML and B2B

With XML, financial information can be exchanged over the Internet.

Expect to see a lot about XML and B2B (Business To Business) in the near future.

XML is going to be the main language for exchanging financial information between businesses over the Internet. A lot of interesting B2B applications are under development.

XML can be used to Share Data

With XML, plain text files can be used to share data.

Since XML data is stored in plain text format, XML provides a software- and hardware-independent way of sharing data.

This makes it much easier to create data that different applications can work with. It also makes it easier to expand or upgrade a system to new operating systems, servers, applications, and new browsers.

XML can be used to Store Data

With XML, plain text files can be used to store data.

XML can also be used to store data in files or in databases. Applications can be written to store and retrieve information from the store, and generic applications can be used to display the data.

XML can make your Data more Useful

With XML, your data is available to more users.

Since XML is independent of hardware, software and application, you can make your data available to other than only standard HTML browsers.

Other clients and applications can access your XML files as data sources, like they are accessing databases. Your data can be made available to all kinds of "reading machines" (agents), and it is easier to make your data available for blind people, or people with other disabilities.

XML can be used to Create new Languages

XML is the mother of WAP and WML.

The Wireless Markup Language (WML), used to markup Internet applications for handheld devices like mobile phones, is written in XML.

If Developers have Sense

If they DO have sense, all future applications will exchange their data in XML.

The future might give us word processors, spreadsheet applications and databases that can read each other's data in a pure text format, without any conversion utilities in between.

We can only pray that Microsoft and all the other software vendors will agree.

XML Syntax

The syntax rules of XML are very simple and very strict. The rules are very easy to learn, and very easy to use.

Because of this, creating software that can read and manipulate XML is very easy to do.

An example XML document

XML documents use a self-describing and simple syntax.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The first line in the document - the XML declaration - defines the XML version and the character encoding used in the document. In this case the document conforms to the 1.0 specification of XML and uses the ISO-8859-1 (Latin-1/West European) character set.

The next line describes the root element of the document (like it was saying: "this document is a note"):

```
<note>
```

The next 4 lines describe 4 child elements of the root (to, from, heading, and body):

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

And finally the last line defines the end of the root element:

```
</note>
```

Can you detect from this example that the XML document contains a Note to Tove from Jani? Don't you agree that XML is pretty self-descriptive?

All XML elements must have a closing tag

With XML, it is illegal to omit the closing tag.

In HTML some elements do not have to have a closing tag. The following code is legal in HTML:

```
<p>This is a paragraph
<p>This is another paragraph
```

In XML all elements must have a closing tag, like this:

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

Note: You might have noticed from the previous example that the XML declaration did not have a closing tag. This is not an error. The declaration is not a part of the XML document itself. It is not an XML element, and it should not have a closing tag.

XML tags are case sensitive

Unlike HTML, XML tags are case sensitive.

With XML, the tag <Letter> is different from the tag <letter>.

Opening and closing tags must therefore be written with the same case:

```
<Message>This is incorrect</message>

<message>This is correct</message>
```

All XML elements must be properly nested

Improper nesting of tags makes no sense to XML.

In HTML some elements can be improperly nested within each other like this:

```
<b><i>This text is bold and italic</b></i>
```

In XML all elements must be properly nested within each other like this:

```
<b><i>This text is bold and italic</i></b>
```

All XML documents must have a root element

All XML documents must contain a single tag pair to define a root element.

All other elements must be within this root element.

All elements can have sub elements (child elements). Sub elements must be correctly nested within their parent element:

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

Attribute values must always be quoted

With XML, it is illegal to omit quotation marks around attribute values.

XML elements can have attributes in name/value pairs just like in HTML. In XML the attribute value must always be quoted. Study the two XML documents below. The first one is incorrect, the second is correct:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note date=12/11/2002>
<to>Tove</to>
<from>Jani</from>
</note>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note date="12/11/2002">
<to>Tove</to>
<from>Jani</from>
</note>
```

The error in the first document is that the date attribute in the note element is not quoted.

This is correct: date="12/11/2002". This is incorrect: date=12/11/2002.

With XML, white space is preserved

With XML, the white space in your document is not truncated.

This is unlike HTML. With HTML, a sentence like this:

Hello my name is Tove,

will be displayed like this:

Hello my name is Tove,

because HTML reduces multiple, consecutive white space characters to a single white space.

With XML, CR / LF is converted to LF

With XML, a new line is always stored as LF.

Do you know what a typewriter is? Well, a typewriter is a mechanical device used in the previous century to produce printed documents. :-)

After you have typed one line of text on a typewriter, you have to manually return the printing carriage to the left margin position and manually feed the paper up one line.

In Windows applications, a new line is normally stored as a pair of characters: carriage return (CR) and line feed (LF). The character pair bears some resemblance to the typewriter actions of setting a new line. In Unix applications, a new line is normally stored as a LF character. Macintosh applications use only a CR character to store a new line.

Comments in XML

The syntax for writing comments in XML is similar to that of HTML.

<!-- This is a comment -->

XML Elements

XML Elements are extensible and they have relationships.

XML Elements have simple naming rules.

XML Elements are Extensible

XML documents can be extended to carry more information.

Look at the following XML NOTE example:

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

Let's imagine that we created an application that extracted the <to>, <from>, and <body> elements from the XML document to produce this output:

MESSAGE

To: Tove
From: Jani

Don't forget me this weekend!

Imagine that the author of the XML document added some extra information to it:

```
<note>
<date>2002-08-01</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Should the application break or crash?

No. The application should still be able to find the <to>, <from>, and <body> elements in the XML document and produce the same output.

XML documents are Extensible.

XML Elements have Relationships

Elements are related as parents and children.

To understand XML terminology, you have to know how relationships between XML elements are named, and how element content is described.

Imagine that this is a description of a book:

My First XML

Introduction to XML

- What is HTML
- What is XML

XML Syntax

- ✍ Elements must have a closing tag
- ✍ Elements must be properly nested

Imagine that this XML document describes the book:

```
<book>
<title>My First XML</title>
<prod id="33-657" media="paper"></prod>
<chapter>Introduction to XML
<para>What is HTML</para>
<para>What is XML</para>
</chapter>

<chapter>XML Syntax
<para>Elements must have a closing tag</para>
<para>Elements must be properly nested</para>
</chapter>

</book>
```

Book is the **root element**. Title, prod, and chapter are **child elements** of book. Book is the **parent element** of title, prod, and chapter. Title, prod, and chapter are **siblings** (or **sister elements**) because they have the same parent.

Elements have Content

Elements can have different content types.

An **XML element** is everything from (including) the element's start tag to (including) the element's end tag.

An element can have **element** content, **mixed** content, **simple** content, or **empty** content. An element can also have **attributes**.

In the example above, book has **element content**, because it contains other elements. Chapter has **mixed content** because it contains both text and other elements. Para has **simple content** (or **text content**) because it contains only text. Prod has **empty content**, because it carries no information.

In the example above only the prod element has **attributes**. The **attribute** named id has the **value** "33-657". The **attribute** named media has the **value** "paper".

Element Naming

XML elements must follow these naming rules:

- ? Names can contain letters, numbers, and other characters
- ? Names must not start with a number or punctuation character

- Names must not start with the letters xml (or XML or Xml ..)
- Names cannot contain spaces

Take care when you "invent" element names and follow these simple rules:

Any name can be used, no words are reserved, but the idea is to make names descriptive. Names with an underscore separator are nice.

Examples: <first_name>, <last_name>.

Avoid "-" and "." in names. For example, if you name something "first-name," it could be a mess if your software tries to subtract name from first. Or if you name something "first.name," your software may think that "name" is a property of the object "first."

Element names can be as long as you like, but don't exaggerate. Names should be short and simple, like this: <book_title> not like this: <the_title_of_the_book>.

XML documents often have a corresponding database, in which fields exist corresponding to elements in the XML document. A good practice is to use the naming rules of your database for the elements in the XML documents.

Non-English letters like éòá are perfectly legal in XML element names, but watch out for problems if your software vendor doesn't support them.

The ":" should not be used in element names because it is reserved to be used for something called namespaces (more later).

XML Attributes

XML elements can have attributes in the start tag, just like HTML.

Attributes are used to provide additional information about elements.

XML Attributes

XML elements can have attributes.

From HTML you will remember this: <IMG SRC="computer.gif">. The SRC attribute provides additional information about the IMG element.

In HTML (and in XML) attributes provide additional information about elements:

```
<img src="computer.gif">
<a href="demo.asp">
```

Attributes often provide information that is not a part of the data. In the example below, the file type is irrelevant to the data, but important to the software that wants to manipulate the element:

```
<file type="gif">computer.gif</file>
```

Quote Styles, "female" or 'female'?

Attribute values must always be enclosed in quotes, but either single or double quotes can be used. For a person's sex, the person tag can be written like this:

```
<person sex="female">
```

or like this:

```
<person sex='female'>
```

Note: If the attribute value itself contains double quotes it is necessary to use single quotes, like in this example:

```
<gangster name='George "Shotgun" Ziegler'>
```

Note: If the attribute value itself contains single quotes it is necessary to use double quotes, like in this example:

```
<gangster name="George 'Shotgun' Ziegler">
```

Use of Elements vs. Attributes

Data can be stored in child elements or in attributes.

Take a look at these examples:

```
<person sex="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<person>
  <sex>female</sex>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

In the first example sex is an attribute. In the last, sex is a child element. Both examples provide the same information.

There are no rules about when to use attributes, and when to use child elements. My experience is that attributes are handy in HTML, but in XML you should try to avoid them. Use child elements if the information feels like data.

My Favorite Way

I like to store data in child elements.

The following three XML documents contain exactly the same information:

A date attribute is used in the first example:

```
<note date="12/11/2002">
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

A date element is used in the second example:

```
<note>
<date>12/11/2002</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

An expanded date element is used in the third: (THIS IS MY FAVORITE):

```
<note>
<date>
  <day>12</day>
  <month>11</month>
  <year>2002</year>
</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Avoid using attributes?

Should you avoid using attributes?

Some of the problems with using attributes are:

- attributes cannot contain multiple values (child elements can)
- attributes are not easily expandable (for future changes)
- attributes cannot describe structures (child elements can)
- attributes are more difficult to manipulate by program code
- attribute values are not easy to test against a Document Type Definition (DTD) - which is used to define the legal elements of an XML document

If you use attributes as containers for data, you end up with documents that are difficult to read and maintain. Try to use **elements** to describe data. Use attributes only to provide information that is not relevant to the data.

Don't end up like this (if you think this looks like XML, you have not understood the point):

```
<note day="12" month="11" year="2002"
to="Tove" from="Jani" heading="Reminder"
body="Don't forget me this weekend!">
</note>
```

An Exception to my Attribute rule

Rules always have exceptions.

My rule about attributes has one exception:

Sometimes I assign ID references to elements. These ID references can be used to access XML elements in much the same way as the NAME or ID attributes in HTML. This example demonstrates this:

```
<messages>
  <note id="p501">
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>

  <note id="p502">
    <to>Jani</to>
    <from>Tove</from>
    <heading>Re: Reminder</heading>
    <body>I will not!</body>
  </note>
</messages>
```

The ID in these examples is just a counter, or a unique identifier, to identify the different notes in the XML file, and not a part of the note data.

What I am trying to say here is that metadata (data about data) should be stored as attributes, and that data itself should be stored as elements.

Viewing XML Files

Raw XML files can be viewed in Mozilla Firefox, IE 5.0+ and in Netscape 6.

However, to make XML documents to display like nice web pages, you will have to add some display information.

Viewing XML Files

In Mozilla Firefox and IE 5.0+:

Click on a link to an XML file, type the direct URL in the address bar, or double-click on the name of an XML file in a folder. The XML document will be displayed with color-coded root and child

elements. A plus (+) or minus sign (-) to the left of the elements can be clicked to expand or collapse the element structure. To view the raw XML source (without the + and - signs), select "View Page Source" or "View Source" from the browser menu.

In Netscape 6:

Open the XML file (via a link or URL), then right-click in XML file and select "View Page Source". The XML document will then be displayed with color-coded root and child elements.

Look at this XML file: note.xml

Note: Do not expect XML files to be formatted like HTML documents!

Viewing an Invalid XML File

If an erroneous XML file is opened, the browser will report the error.

Look at this XML file: note_error.xml

Displaying XML with CSS

With CSS (Cascading Style Sheets) you can add display information to an XML document.

Displaying your future XML files with CSS?

Would you use CSS to format your future XML files? No, we don't think so! But we could not resist giving it a try:

Take a look at this XML file: The CD catalog

Then look at this style sheet: The CSS file

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

Below is a fraction of the XML file. The second line, **<?xml-stylesheet type="text/css" href="cd_catalog.css"?>**, links the XML file to the CSS file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="cd_catalog.css"?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD>
```

```
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
.
.
.
.
</CATALOG>
```

We DO NOT believe that formatting XML with CSS is the future of the Web. Even if it looks right to use CSS this way, we DO believe that formatting with XSL will be the new standard (as soon as all main browsers support it).

Displaying XML with XSL

With XSL you can add display information to your XML document.

Displaying XML with XSL

XSL is the preferred style sheet language of XML.

XSL (the eXtensible Stylesheet Language) is far more sophisticated than CSS. One way to use XSL is to transform XML into HTML before it is displayed by the browser as demonstrated in these examples:

If you have Netscape 6 or IE 5 or higher you can view the XML file and the XSL style sheet.

Below is a fraction of the XML file, with an added XSL reference. The second line, **<?xml-stylesheet type="text/xsl" href="simple.xsl"?>**, links the XML file to the XSL file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="simple.xsl"?>
<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>
       two of our famous Belgian Waffles
    </description>
    <calories>650</calories>
  </food>
</breakfast_menu>
```

XML in Data Islands

With Internet Explorer 5.0 and higher, XML can be embedded within HTML pages in Data Islands.

XML Embedded in HTML

The unofficial <xml> tag is used to embed XML data within HTML.

XML data can be embedded directly in an HTML page like this:

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

Or a separate XML file can be embedded like this:

```
<xml id="note" src="note.xml">
</xml>
```

Note that the <xml> tag is an HTML element, not an XML element.

Data Binding

Data Islands can be bound to HTML elements (like HTML tables).

In the example below, an XML Data Island with an ID "cdcat" is loaded from an external XML file. An HTML table is bound to the Data Island with a data source attribute, and finally the tabledata elements are bound to the XML data with a data field attribute inside a span.

```
<html>
<body>

<xml id="cdcat" src="cd_catalog.xml"></xml>

<table border="1" datasrc="#cdcat">
<tr>
<td><span datafld="ARTIST"></span></td>
<td><span datafld="TITLE"></span></td>
</tr>
</table>

</body>
</html>
```

Example

```
<html>
<body>

<xml id="cdcat" src="cd_catalog.xml"></xml>

<table border="1" datasrc="#cdcat">

<thead>
<tr><th>Artist</th><th>Title</th></tr>
</thead>

<tfoot>
<tr><th colspan="2">This is my CD collection</th></tr>
</tfoot>

<tbody>
<tr>
<td><span datafld="artist"></span></td>
<td><span datafld="title"></span></td>
</tr>
</tbody>

</table>

</body>
</html>
```

The Microsoft XML Parser

To read and update - create and manipulate - an XML document, you need an XML parser.

Using the XML parser

The Microsoft XML parser comes with Microsoft Internet Explorer 5.0.

Once you have installed IE 5.0, the parser is available to scripts, both inside HTML documents and inside ASP files. The parser features a language-neutral programming model that supports:

- JavaScript, VBScript, Perl, VB, Java, C++ and more
- W3C XML 1.0 and XML DOM
- DTD and validation

If you are using JavaScript in IE 5.0, you can create an XML document object with the following code:

```
var xmlDoc=new ActiveXObject("Microsoft.XMLDOM")
```

If you are using VBScript, you create the XML document object with the following code:

```
set xmlDoc=CreateObject("Microsoft.XMLDOM")
```

If you are using VBScript in ASP, you can use the following code:

```
set xmlDoc=Server.CreateObject("Microsoft.XMLDOM")
```

Loading an XML file into the parser

XML files can be loaded into the parser using script code.

The following code loads an XML document (note.xml) into the XML parser:

```
<script type="text/javascript">
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.load("note.xml")
// ....... processing the document goes here
</script>
```

The second line in the code above creates an instance of the Microsoft XML parser.

The third line turns off asynchronized loading, to make sure that the parser will not continue execution before the document is fully loaded.

The fourth line tells the parser to load the XML document called note.xml.

Loading pure XML text into the parser

XML text can also be loaded from a text string.

The following code loads a text string into the XML parser:

```
<script type="text/javascript">

var text="<note>"
text=text+"<to>Tove</to><from>Jani</from>"
text=text+"<heading>Reminder</heading>"
text=text+"<body>Don't forget me this weekend!</body>"
text=text+"</note>"

var xmlDoc = new ActiveXObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.loadXML(text)
// ....... processing the document goes here
</script>
```

Note that the "loadXML" method (instead of the "load" method) is used to load a text string.

Displaying XML with JavaScript

To display XML you can use JavaScript.

JavaScript (or VBScript) can be used to import data from an XML file and display the XML data inside an HTML page.

To see how XML and HTML complement each other this way; first look at the **XML** document (note.xml), then open the **HTML** document (note.htm) that contains a JavaScript which reads the XML file and displays the information inside predefined spans in the HTML page.

note.xml

```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
- <!--  Edited with XML Spy v4.2
  -->
- <note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
  </note>
```

note.htm

```
<html>
<head>
<script
type="text/javascript"
for="window"
event="onload">

var xmlDoc=new ActiveXObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.load("note.xml")

nodes=xmlDoc.documentElement.childNodes
a1.innerText=nodes.item(0).text
a2.innerText=nodes.item(1).text
a3.innerText=nodes.item(2).text
a4.innerText=nodes.item(3).text

</script>
</head>

<body bgcolor="yellow">
```

```
<h1>W3Schools Internal Note</h1>

<b>To:</b>
<span id="a1"></span>
<br />

<b>From:</b>
<span id="a2"></span>
<hr />

<b><span id="a3"></span></b>

<hr />
<span id="a4"></span>

</body>
</html>
```

XML in Real Life

Some real-life examples of how XML can be used to carry information.

Example: XML News

XMLNews is a specification for exchanging news and other information.

Using such a standard makes it easier for both news producers and news consumers to produce, receive, and archive any kind of news information across different hardware, software, and programming languages.

An example XML News document:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<nitf>

<head>
<title>Colombia Earthquake</title>
</head>

<body>

<body.head>
<headline>
<hl1>143 Dead in Colombia Earthquake</hl1>
</headline>
<byline>
<bytag>By Jared Kotler, Associated Press Writer</bytag>
</byline>
```

```
<dateline>
<location>Bogota, Colombia</location>
<story.date>Monday January 25 1999 7:28 ET</story.date>
</dateline>
</body.head>

</body>

</nitf>
```