

XML

OVERVIEW

XML is one of those boring ideas that becomes exciting when it makes businesses run more smoothly. Really it's just some general rules for how to write down information so that computers as well as people can read it – mainly computers, though. It's not even a full set of rules; it's just enough to help people make a start on designing their own formats for sharing information.

So in publishing, people have taken the XML rules and added some extra, book-specific ones until they came up with a standardised way for describing new titles – or old ones, for that matter. The standardisation makes it easy for the whole industry to share information. Publishers can tell all the book stores, book clubs, library services and industry databases about a new title by sending them all a copy of the same XML file.

In other sectors, XML is being used as the starting point for storing a multitude of different types of information. There are XML standards for creating an invoice or describing a gene – or (to return to publishing) to store the contents of an e-book in a standardised way.

THE GUTS OF XML (MADE PALATABLE FOR NON-TECHIES)

The first thing to learn about XML is good news: it's written in English. Or rather it's written using ordinary words, with a few squiggles added, and not in some sort of computer hieroglyphics.

So, imagine you're sending out details of a new book you're publishing. Naturally you want everyone to add its details to their stock systems so they can easily order it. Let's start with the title, *The Life and Times of Ned Lud*. A human being can take a guess that it's a book title, but in XML you always label information to make it clear. So we might write this:

```
<TitleText>The Life and Times of Ned Lud</TitleText>
```

It's like that maxim for lecturing: tell people what you're going to say, then say it, then tell them what you just said. So that line above says: Here comes something called TitleText, 'The Life and Times of Ned Lud', that's the end of the TitleText. When you put '/' in front of a label you're marking the end of something.

Of course you can make up your own names for information. You could choose <NameOfBook> or <FieldDD6_Alpha_Gobbledygook>, but it just so happens that <TitleText> is the name that's been agreed on by largish group of book publishers as

part of the ONIX standard.

The bit in the angle brackets is called a 'tag'. It's fairly easy to see why; you 'tag' information to say what it means. For instance, if all you have is the piece of text 'Winston Churchill', it's difficult to tell whether that's a book about him or a book written by him – both are plausible. So everything gets tagged for clarity's sake. Take a look at this:

```
<Author>Katie Daynes</Author>
<TitleText>Winston Churchill</TitleText>
<ISBN>074606814X</ISBN>
<Publisher>Usborne Publishing Ltd</Publisher>
```

Hopefully it's pretty obvious what that all means. Each 'tag' has a start and an end and the bit in the middle is the information you want to share – also known as the 'contents' of the tag. A computer can easily read it and so can a person (with a little effort). Unfortunately for us, information about books gets complicated and so the XML used to store it has to get complicated too.

For instance, what do we do if there's more than one author? Or if there's no author, just an editor and some contributors? And what about all the other pieces of information we might want to share, like publication date, price and distributor details? The people who wrote Onix came up with something that's not exactly perfect, but it does allow you store a vast amount of information about a title in a structured way.

One feature of XML the Onix people made use of was the idea of putting one tag inside another. I want to show you what that looks like, but real Onix documents are a bit difficult to read, so this next example is just a made-up one; it doesn't follow the Onix standard. But on the plus side, it's actually possible for a human to understand it.

```
<Book>
<Title>
<MainTitle>The Life and Times of Ned Lud</MainTitle>
<SubTitle>Backward Looking Visionary</SubTitle>
</Title>
<Author>
<FirstName>Emma</FirstName>
<Surname>Barnes</Surname>
</Author>
<Illustrator>
<FirstName>Rob</FirstName>
<Surname>Jones</Surname>
</Illustrator>
</Book>
```

So in this made-up example, if a <Surname> tag is inside an <Author> tag, then it's the name of an Author; if it's inside an <Illustrator> tag, then it's the name of an illustrator. Easy.

Amazingly, we're nearly finished, but there's another XML feature we need to cover first. It's called an attribute. The idea of an attribute is to let you add some additional information about a tag. Let's imagine no one liked the last excerpt so we went away and came up with a new XML format for describing books and it looked like this:

```
<Book>
<Title type='Main'> The Life and Times of Ned Lud</Title>
<Title type='Subtitle'>Backward Looking Visionary</Title>
<Person role='Author'>Emma Barnes</Person>
<Person role='Illustrator'>Rob Jones</Person>
</Book>
```

Again, it's not what the real Onix standard looks like; Onix is rather confusing. This is just a made up format. It conveys pretty much the same information as the last excerpt, but instead of putting one tag inside another to spell out its meaning we're adding extra info inside the angle brackets.

XML gives you the scope to choose either of them as your standard; it's up to you which you prefer.

OK. So there's lots more you could learn about XML, but that's enough so you can join in fun dinner party conversations on the subject and look techies in the eye without flinching. Let's get back to the real world of publishing.

THE ONIX STANDARD

If you want an easy way to tell Nielsen or Amazon or Waterstone's about a new book, you can put all the relevant info in an Onix message and e-mail it to them. In case you're interested, the British contributors to the Onix standard were the BIC, made up of the Library Association, The British Library, The Booksellers Association and The Publishers Association, so it's got some weight behind it.

It's a gigantic and complicated standard because it needs to be able to hold gigantic and complicated amounts of information for each title. For instance, it gives you tags for listing the back cover quotes on your book, and giving the names of each quote contributor and the organisations they work for. It lets you include details of discounts and promotions by date and region. It holds information on formats and rights and physical dimensions – and even what units the measurements are being given in.

Unfortunately for anyone who wants to open an Onix message and actually read the

contents, the standard also makes use of numbers where a name would have been easier to read. For instance, if you want to know whether a book has been published you could look at the `<PublishingStatus>` tag. Here's one:

```
<PublishingStatus>04</PublishingStatus>
```

But what does '04' mean? Well, if you hunt down a copy of the Onix documentation you'll find this list:

List 64: Publishing status

00 Unspecified Status is not specified (as distinct from unknown): the default if the `<PublishingStatus>` element is not sent. Also to be used in applications where the element is considered mandatory, but the sender of the ONIX message chooses not to pass on status information.

01 Cancelled The product was announced, and subsequently abandoned; the `<PublicationDate>` element must not be sent.

02 Forthcoming Not yet published, must be accompanied by expected date in `<PublicationDate>`.

03 Postponed indefinitely The product was announced, and subsequently postponed with no expected publication date; the `<Publication Date>` element must not be sent.

04 Active The product was published, and is still active in the sense that the publisher will accept orders for it, though it may or may not be immediately available, for which see `<SupplyDetail>`.

05 No longer our product Ownership of the product has been transferred to another publisher (with details of acquiring publisher if possible in PR.19).

06 Out of stock indefinitely The product was active, but is now inactive in the sense that (a) the publisher will not accept orders for it, though stock may still be available elsewhere in the supply chain, and (b) there are no current plans to bring it back into stock. Code 06 does not specifically imply that returns are or are not still accepted.

07 Out of print The product was active, but is now permanently inactive in the sense that (a) the publisher will not accept orders for it, though stock may still be available elsewhere in the supply chain, and (b) the product will not be made available again under the same ISBN.

Code 07 normally implies that the publisher will not accept returns beyond a specified date.

08 Inactive The product was active, but is now permanently or indefinitely inactive in the sense that the publisher will not accept orders for it, though stock may still be available elsewhere in the supply chain.

Code 08 covers both of codes 06 and 07, and may be used where the distinction between those values is either unnecessary or meaningless.

09 Unknown The sender of the ONIX record does not know the current publishing status.

10 Remaindered The product is no longer available from the current publisher, under the current ISBN, at the current price. It may be available to be traded through another channel.

That's the contents of List 64, the document which explains the meaning of the numbers in a <PublishingStatus> tag. There are 102 of these lists explaining what the various different numbers and codes mean, so while humans can get the gist of what's in an Onix message, the details are often hard to follow. The Onix people could have chosen to use the words 'Active' and 'Remaindered' instead of the numbers '04' and '10', but they probably took the view that machines, rather than people, would be reading these messages and numbers were more concise.

HOW CAN I USE ONIX?

Having established that Onix messages are swines to read – unless you're a machine – the obvious thing to do is enlist the help of a machine whenever you want to work with Onix.

You can buy software from a variety of companies to make up your Onix messages. Instead of us looking up what all the tags and numbers mean, the software does that. What we humans see are screens that look like this:

The program gives you helpful forms to fill out; they list the available options in words. Then the program inserts the relevant code on your behalf. So if you choose 'Paperback' from the drop-down list, the program puts the code 'BC' into your Onix message, saving you the bother of looking it up. (If we actually had to write Onix messages by hand, we probably wouldn't bother.)

WHAT ELSE CAN YOU DO WITH ONIX?

One thing you can do with your Onix messages is to use them to create part of your website. Since you go to all the trouble of typing in a book's details whenever you want to tell the world about a new title, you might as well make use of that same info to create a page in your online catalogue. The sneaky benefit of doing this (apart from saving time) is that if you've made a mistake when you typed in the book's details, it's so much easier to proof a nicely-formatted webpage than a database.

How do you turn an Onix message into a web-page? Well it's a bit techie, but the idea is you make something called an XSL template that reads in Onix XML and spits out HTML (which is the code websites are written in, in case you didn't know). You only have to write the XSL template once, and then any time you update a book's details or add a new title, you just run it through the template and out pops an updated web-page.

Whether or not you want to link your book data to your website like that, the point is that once a book's details are captured in an industry standard format, you're well set

up to use that information in other ways, for instance to create catalogues, AIs, flyers, retailer spreadsheets... in fact any time you use bibliographic data.

OTHER USES FOR XML

Of course Onix is just one flavour of XML. There are XML standards for almost any kind of information you can imagine. Not only are Microsoft Word documents now XML-based but so are a large proportion of the web pages on the Internet. If you want to know more about that, read on.

XML AND CSS

Although this section isn't specifically about publishing, there's a philosophy behind using XML on the web that publishers might sympathise with. It's based on the idea that presentation and content are two different things. But what does that actually mean? Well, the idea is that you can label a piece of text as a title without specifying the font-size or typeface or colour it should be displayed in.

Keeping presentation separate from meaning opens up a number of technical possibilities and can make a big difference to efficiency if you have a lot of information to manage. We've touched on this issue already, but to illustrate, imagine that you're publishing a series of little-known biographies. Now look at this:

```
<p><Bold>David Smith</Bold></p>
<p>By</p>
<p><Italic>Peter Smith</Italic></p>
```

That's the sort of thing you might see if you look at the code behind an ordinary web page. (The <p> tags just mark the starts and ends of paragraphs, telling the web browser when to start a new line.)

By noticing the word 'by' a human can figure out that the second name is the author and the first is probably the book title.

Compare it with this:

```
<p class="BookTitle">David Smith</p>
<p>By</p>
<p class="Author">Peter Smith</p>
p.Author {
font: bold
color: 'Blue';
}
p.BookTitle {
font: italic
```

```
color: 'Blue';  
}
```

Two things are different in the second example. We've added attributes to the information to say what it is rather than how to display it. Then underneath we've added something called a 'stylesheet' which will specify how to display the tags we used above. The end result on screen might look the same but the page is easier for machines to extract meaning from and it's also easier for designers to maintain.

If web pages had properly descriptive XML-style tags in them, Google could find you information about Tom Jones the book without accidentally including any references to Tom Jones the singer. In effect it could organise the Internet like a library rather than a car boot sale. Search engines that 'understand' some of the pages they read could make the web a much more useful resource for everyone. No, really.

In that second example above, we added a 'stylesheet' (the bit in bluey grey). Stylesheets bring together all the presentation information, keeping it separate from the content, and making it easier to revise the look of a document. Better still, you can use one stylesheet to control the presentation of web pages covering thousands of different books. Update that one stylesheet and all the web pages change, sharing the same consistent layout. If you mix the presentation information (like what's in bold, what's in red, etc;) in with the words, then you'd have to update each of those thousands of pages individually to change the look of your site.

Designers get excited about that sort of thing. In fact there's a web-site called 'css Zen Garden' where a single web page has had hundreds of different alternate stylesheets written for it. Each version is displaying the same underlying information, only the aesthetics have changed, but the transformation is amazing. Take a look here.

Most web designers have been using CSS to control the look of their web-sites for the last few years, but badly tagged text makes that more difficult. And as for machines figuring out what any given web page is about, that project is still in its infancy. The next step in both of those arenas is to properly tag up and organise the underlying content on web sites. Once that happens, it'll be like 1999 all over again, with Internet companies hyperventilating over the, like, awesome possibilities and the stock market losing its mind over anything relating to web technology. And unlike 1999, some of it might work.