

# How To Guides for XML

## "How To" Guides for XML

This is an automatically-generated printer-friendly PDF version of the HTML web page at [How To Guides for XML](#) ← CLICK HERE to get the standard HTML version of this document.

## How To Look At XML

### XML paradigms compared to programming paradigms

In the same way that there is more than one paradigm in programming, there is more than one way to look at XML. As for the former, there is procedural, Object-Oriented Programming ([OOP](#)) and Aspect-Oriented Programming ([AOP](#)). In the same manner, there are a number of ways to look at XML:

#### as simply a way to store and/or transmit data

XML can be viewed as yet another format for storing data items. In some ways, this is equivalent to doing procedural programming in an object-oriented programming language. The advantage is that you can get the immediate job done fairly well without having to shift your paradigm. The disadvantage is that procedure code is typically not reusable. So the next time a similar task needs to be accomplished, new code must be developed. This is called [Reinventing The Wheel \(RTW\)](#).

Likewise, simply looking at XML as a way to store and/or transmit data can make it easier to implement a specific application. An example of this is [RSS 0.9x / 2.0](#), which makes good use of XML as a way to transmit data items to accomplish the job of delivering syndicated web feeds. The limitations of this might not be apparent until you start encountering problems with the schema or, such as in the case of RSS, the lack of one. Because of this, RSS feeds cannot be validated with standard XML validators; it becomes necessary to "reinvent the wheel" with validation specific to the RSS format. The lack of a deterministic schema is one of the [limitations of RSS](#) and is not even resolved with the [Atom Syndication Format](#), although [Open Feed Format](#) does provide one and can be easily transformed into either RSS or Atom Syndication Formats.

#### as a way to store and/or aggregate content

It takes a serious paradigm shift to start looking at XML as a way to store content, rather than just collections of data items. This is equivalent to the shift in paradigms from looking at an object-oriented programming language as a set of procedural instructions to a way to manipulate more robust objects.

This is the biggest jump that is needed in making good use of XML, just like understanding objects is the biggest jump in understanding OOP. When you see code written in an OOP language that is consistently passing a particular object as a parameter to almost every method, you recognize that the author of that code has not yet made the leap. Likewise, when you look at the design of the RSS feed format, you can see that the author(s) had not yet made the leap to the next level, (or possibly that they did not want to rely on users having made the leap).

This does not mean that you have to abandon [HTML](#) development. On second thought, I take that back - actually, yes it does. It means you have to abandon 1997 [HTML 4.x](#) development and IE 4.x compatibility in favor of [XHTML](#) and more recent browsers. [XHTML](#) web pages are simply a particular type of XML documents. Therefore, you should start to think of [XHTML](#) not just as a markup language for creating web pages, but as a means for delivering content. Once you've made this leap to the new paradigm, you can do things like:

- separate the content of your web site from the presentation
- make better use of content by tailoring the presentation of the content for specific delivery channels; an example would be for users of handheld devices such as cell phones and [PDAs](#). See [Sample .mobi](#) for a good example of this.
- move common elements, those things that provide the [look and feel \(LaF\)](#) of a web site, into templates, which, because they can be cached by most browsers, reduces bandwidth requirements and improves page load times

#### as a way to integrate content

Aspect-Oriented Programming takes [OOP](#) to the next level by looking at objects in multiple dimensions, rather than just a single dimension. The analogy with the three-dimensional space, of which we are quite familiar, would be that OOP looks at objects in the vertical (up/down, class/subclass) direction, and AOP adds the horizontal directions. The view of objects in the horizontal directions are called cross-cutting concerns.

Likewise, if you've already made the leap to looking at [XML](#) as a way to store and/or aggregate content, it's relatively easy to take this to the next level and consider XML as a means for integration of content. Content and navigation do not need to be limited to simply moving through web pages in sequence; it can simultaneously be gathered from multiple directions and integrated. This was the promise of the [Semantic Web](#), but has been painfully slow in becoming reality, presumably because most content providers have not yet even made the first leap in making good use of XML in their content.

# How To Guides for XML

## Using XSL Templates

### How to repurpose mobile content for traditional web browsers

When content needs to be delivered to both traditional web browsers and smaller mobile devices, one technique for producing content for both channels from the same source is to code the content for one channel and [repurpose](#) it for the other channel. In most cases, the content for traditional web browsers already exists, so the tendency is to try to repurpose that content for smaller devices.

However, it's much easier to design the content for the smaller portable devices and use XSL templates to repurpose the content for larger displays by wrapping it with additional markup code.

This technique is particularly useful for sites using a ".mobi" domain name managed by the dotMobi registry, since the registry [requires sites to be designed for mobile devices](#). For an example see the [Sample .mobi site](#)

### Sample XHTML file

The [XHTML](#) file contains the page-specific content that is delivered to all types of browsers. No site-wide common elements are included, since those are put into separate templates.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="/sample.xsl"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
"http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!-- Copyright (c) 2006 Accilent Corp. (http://www.Accilent.com/) for How To Guides
(http://www.HowToGuides.com/). Provided AS IS; no warranties are expressed or implied. Permission is hereby
granted to use and/or modify. -->
<head>
<link rel="stylesheet" type="text/css" href="/screen.css" media="screen"/>
<link rel="stylesheet" type="text/css" href="/handheld.css" media="handheld"/>
<title>Sample .mobi site</title>
</head>
<body>
<h1></h1>
<h2>Home Page</h2>
<p>This is the actual content that shows up on small mobile devices. The content can include <a
href="http://www.Acronyms.net/terms/h/Hypertext-Transfer-Protocol/" title="Hypertext Transfer
Protocol"><abbr>HTTP</abbr></a> links to other web pages, such as:</p>
<p><a href="/detect.cgi?test" accesskey="d">Detect browser type</a></p>
<p>as well as specialized links for mobile devices with phone functions, such as:</p>
<p><a href="wtai://wp/mc;14123373113" accesskey="w">Call the webmaster</a></p>
</body>
</html>
```

See [.mobi sites in Internet Explorer](#) for issues displaying ".mobi" web sites in an IE browser.

### Sample XSL templates

Common elements that need to appear on every page throughout the site would be put into templates in an XSL file referenced by the [XHTML](#) pages.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- uncomment to validate XSLT
<!DOCTYPE xsl:transform SYSTEM "http://www.sample.mobi/dtd/xslt.dtd">
-->
<xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:html="http://www.w3.org/1999/xhtml" xmlns="http://www.w3.org/1999/xhtml" exclude-result-prefixes="html">
<!-- Copyright (c) 2006 Accilent Corp. (http://www.Accilent.com/) for How To Guides
(http://www.HowToGuides.com/). Provided AS IS; no warranties are expressed or implied. Permission is hereby
granted to use and/or modify. -->
<xsl:output method="xml" version="1.0" omit-xml-declaration="no" indent="yes"
doctype-system="http://www.sample.mobi/dtd/xhtml11-strict.dtd" doctype-public="-//W3C//DTD XHTML 1.0
Strict//EN"/>
<xsl:template match="/">
<xsl:apply-templates select="*" />
</xsl:template>
<xsl:template match="html:head">
<xsl:copy>
```

# How To Guides for XML

```
<xsl:apply-templates select="@*" />
<link rel="stylesheet" type="text/css" href="/screen.css" media="screen" />
<link rel="stylesheet" type="text/css" href="/handheld.css" media="handheld" />
<xsl:apply-templates select="node()" />
</xsl:copy>
</xsl:template>
<xsl:template match="html:body">
<xsl:copy>
<xsl:apply-templates select="@*" />
<h1 style="display:inline">Sample .mobi site</h1>
&#160;a working example of repurposing mobile content as explained by&#160;
<a href="http://www.HowToGuides.com/internet/xml/templates.xml#mobile"></a>
<p style="color: gray">This is a sample showing how templates can be used to make a web site that has been
designed for mobile devices look good in a normal web browser. The important content, which you can do "View
Source" to see, is put into <a href="http://www.Acronyms.net/terms/e/Extensible-Hypertext-Markup-Language/"
title="Extensible Hypertext Markup Language"><abbr>XHTML</abbr></a> pages that are delivered to mobile
devices, and any additional layout and styling for traditional browsers is put into templates and style sheets. Since the
templates and style sheets can be reused for all pages on the site and are cached separately, this has the added
advantage of reducing bandwidth requirements by as much as one half or more.</p>
<hr />
<xsl:apply-templates select="node()" />
<hr />
<p>Click on the following link to verify that this page conforms to the <a
href="http://www.Acronyms.net/terms/e/Extensible-Hypertext-Markup-Language/" title="Extensible Hypertext Markup
Language"><abbr>XHTML</abbr></a> Mobile Profile as required by the dotMobi registry Switch On! guidelines:</p>
<p>
<a href="http://validator.w3.org/check/referer" target="w3cvalidator">
</a>
&#160;
<a href="http://ready.mobi/start.jsp?uri=sample.mobi">MobiReady Report</a>
</p>
<p>The "Mandatory Registrant Rules" for .mobi domains states:</p>
<p style="margin: 0 3em">The "Mandatory Registrant Rules" for .mobi domains states that "Requests for URIs
consisting only of "example.mobi" or "www.example.mobi" must result in a response that is encoded in a format the
device supports or valid <a href="http://www.Acronyms.net/terms/e/Extensible-Hypertext-Markup-Language/"
title="Extensible Hypertext Markup Language"><abbr>XHTML</abbr></a>-Mobile Profile 1.0 or later released
version, where "example" stands for any domain name.</p>
<p>For a site that satisfies these registration requirements for a .mobi domain, the W3C Validator will display:</p>
<p style="margin: 0 3em; background-color: #55b05a; color: white; text-align: center; font-family: Bitstream Vera
Sans, sans-serif; font-size-adjust: .53; font-size: 1.5em; font-weight: 500">This Page Is Valid -//WAPFORUM//DTD
XHTML Mobile 1.0//EN!</p>
<p>Copyright &#169; 2006 Sample .mobi. All rights reserved.</p>
</xsl:copy>
</xsl:template>
<xsl:template match="@*|node()">
<xsl:copy>
<xsl:apply-templates select="@*|node()" />
</xsl:copy>
</xsl:template>
</xsl:transform>
```

More sophisticated sites could have various groups of pages which are similar to each other, but are different from the pages in other groups. In this case, more than one level of templates would be used. For each group of similar pages, a separate template would be created for the common elements on the pages within that group. Those templates would all import a single site-wide template containing the common elements that would appear on all pages across the entire site.

## Style sheet for traditional web browsers

```
img {
border: 0;
}
```

## Style sheet for handheld devices

```
img {
border: 0;
```

# How To Guides for XML

}

## .htaccess file

To deliver the [XHTML](#) and [WML](#) pages with the correct [MIME](#) types, you will need an .htaccess file with the following directives, or their equivalent for the web server software you are using:

```
DirectoryIndex index.xml
```

```
ErrorDocument 404 /pagenotfound.xml
```

```
AddType application/xhtml+xml;charset=utf-8 xml xhtml
```

```
AddType text/vnd.wap.wml;charset=iso-8859-1 wml
```

```
RewriteEngine on
```

```
RewriteCond %{HTTP_USER_AGENT} \ (compatible;[:space:]]*MSIE[:space:]]+[:digit:]]+\.[[:digit:]][:alnum:]]*;
```

```
RewriteRule ^$ http://%{HTTP_HOST}/index.xml [L,R=temp]
```

Contrary to the HTTP standards, Internet Explorer pays more attention to the file extension than to the [MIME](#) types specified by the server. Therefore, the default DirectoryIndex document has a .xml extension. See [.mobi sites in Internet Explorer](#) for more information on the issues with displaying the default DirectoryIndex document in IE.

This is an automatically-generated printer-friendly PDF version of the HTML web pages at [How To Guides for XML](#) ← [CLICK HERE](#) to get the standard HTML version of this document.

