

The Apache Forrest xdocs document-v2.0 DTD

NOTICE: The content of this document doesn't make any sense at all.

This is a demonstration document using all possible elements in the current Apache Forrest xdocs document-v20.dtd

Table of contents

1 Sample Content.....	2
1.1 Block and inline elements.....	2
1.2 Various presentation formats.....	3
1.3 Using sections.....	4
1.4 Sections, the sequel.....	4
1.5 Showing preformatted source code.....	4
1.6 Using tables.....	4
1.7 Using figures.....	5
1.8 Using class attribute on links.....	5
2 DTD changes.....	5
2.1 Changes between document-v13 and document-v20.....	5
2.2 Changes between document-v12 and document-v13.....	5
2.3 Changes between document-v11 and document-v12.....	5

Note:

This is a demonstration document using all possible elements in the current Apache Forrest xdocs document-v2.0.dtd (See the [DTD changes](#) section at the bottom.)

1. Sample Content

Hint: See the xml source to see how the various elements are used and see the [DTD documentation](#).

1.1. Block and inline elements

This is a simple paragraph. Most documents contain a fair amount of paragraphs. Paragraphs are called `<p>`.

With the `<p xml:space="preserve">` attribute, you can declare that whitespace should be preserved, without implying it is in any other way special.

This next paragraph has a class attribute of 'quote'. CSS can be used to present this `<p class='quote'>` in a different style than the other paragraphs. The handling of this quoted paragraph is defined in the `<extra-css>` element in the `skinconf.xml`.

Anyway, like I was sayin', shrimp is the fruit of the sea. You can barbecue it, boil it, broil it, bake it, sautee it. Dey's uh, shrimp-kabobs, shrimp creole, shrimp gumbo. Pan fried, deep fried, stir-fried. There's pineapple shrimp, lemon shrimp, coconut shrimp, pepper shrimp, shrimp soup, shrimp stew, shrimp salad, shrimp and potatoes, shrimp burger, shrimp sandwich. That- that's about it.

A number of in-line elements are available in the DTD, we will show them inside an unordered list (``):

- Here is a simple list item (``).
- Have you seen the use of the `<code>` element in the previous item?
- Also, we have `<sub>` and `<sup>` elements to show content above or below the text baseline.
- There is a facility to *emphasize* certain words using the `` **``** elements.
- We can use `<icon>`s too.
- Another possibility is the `` element:

another feather

, which offers the ability to refer to an image map.
- We have elements for hyperlinking:
 - ``**
Use this to [link](#) to another document. As per normal, this will open the new document in the same browser window.
 - ``**
Use this to [link](#) to the named anchor in the current document.
 - ``**
Use this to [link](#) to another document and go to the named anchor. This will open the new document in the same browser window.**Targetted window control with jump and fork.**
 See demonstration [using class attribute on links](#).
- Oh, by the way, a definition list `<dl>` was used inside the previous list item. We could put another
 - unordered list
 - inside the list item

Or even tables..

inside tables..

or inside lists, but I believe this liberty gets quickly quite hairy as you see.	
--	--

Table 1: A sample nested table

So far for the in-line elements, let's look at some paragraph-level elements.

FIXME (SN):

The `<fixme>` element is used for stuff which still needs work. Mind the `author` attribute!

Note:

Use the `<note>` element to draw attention to something, e.g. ...The `<code>` element is used when the author can't express himself clearly using normal sentences ;-)

Warning:

Sleep deprivation can be the result of being involved in an open source project. (a.k.a. the `<warning>` element).

Important

If you want your own labels for notes and warnings, specify them using the `label` attribute.

Apart from unordered lists, we have ordered lists too, of course.

1. Item 1
2. Item 2
3. This should be 3 if my math is still OK.

1.2. Various presentation formats

This sample document, written in document-v20 XML can be presented via Forrest in a number of different formats. The links in the following list show this document in each of the currently available formats.

Each of the formats can be made available as a link near the top of the page. Actual placement of those links depends on the skin currently in use. Those links are enabled in the `skinconf.xml` via the `<disable-XXX-link>` elements in the `skinconf.xml`

Presentation Format	Description	skinconf.xml Element
HTML	This document in HTML format.	Always generated by default. Cannot be turned off.
XML	This document in its raw XML format.	<code><disable-xml-link></code> . By default, set to true, meaning that this link will not be shown.
PDF	This document as Adobe PDF	<code><disable-pdf-link></code> . By default, set to false, meaning that this link will be shown.
Text	This document as straight text.	<code><disable-txt-link></code> . By default, set to true, meaning that this link will not be shown. NOT YET IMPLEMENTED.
POD	This document as Perl POD (Plain Old Documentation).	<code><disable-pod-link></code> . By default, set to true, meaning that this

	Text with minimal formatting directives. If on a *nix system with perl installed, see "man perlpod".	link will not be shown.
--	--	-------------------------

1.3. Using sections

You can use sections to put some structure in your document. For some strange historical reason, the section title is an attribute of the <section> element.

1.4. Sections, the sequel

Just some second section.

1.4.1. Section 2.1

Which contains a subsection (2.1).

1.5. Showing preformatted source code

Enough about these sections. Let's have a look at more interesting elements, <source> for instance:

```
// This example is from the book _Java in a Nutshell_ by David Flanagan.
// Written by David Flanagan. Copyright (c) 1996 O'Reilly & Associates.
// You may study, use, modify, and distribute this example for any purpose.
// This example is provided WITHOUT WARRANTY either expressed or implied.

import java.applet.*;    // Don't forget these import statements!
import java.awt.*;

public class FirstApplet extends Applet {
    // This method displays the applet.
    // The Graphics class is how you do all drawing in Java.
    public void paint(Graphics g) {
        g.drawString("Hello World", 25, 50);
    }
}
```

CDATA sections are used within <source> elements so that you can write pointy brackets without needing to escape them with messy < entities ...

```
<pointy>
  easy
</pointy>
```

Please take care to still use a sensible line-length within your source elements.

1.6. Using tables

And now for a table:

heading cell 1		heading cell 2		heading cell 3	
data cell		this data cell spans two columns			
Tables can be nested:		column 1		<ul style="list-style-type: none">• and can include most other elements	column 2
		cell A	<ul style="list-style-type: none">• such as lists	cell B	

Table 1: Table caption

1.7. Using figures

And a `<figure>` to end all of this. Note that this can also be implemented with an `` element.

The fine Forrest logo

1.8. Using class attribute on links

The document-v13 had elements `<fork>` and `<jump>`. In document-v20, those elements no longer exist but the functionality can be duplicated by using the `@class` attribute. Even though the opening of separate windows should be under the control of the user, these techniques can still be employed.

Document V1.3	Document V2.0
<code><fork href="faq.html"></code>	<code></code>
<code><jump href="faq.html"></code>	<code></code>

2. DTD changes

See the generated [DTD documentation](#).

2.1. Changes between document-v13 and document-v20

- Renamed `<link>` to `<a>`
- Removed `<fork>` and `<jump>` in favour of the `<a>` element. See demonstration [using class attribute on links](#).

2.2. Changes between document-v12 and document-v13

All v1.2 docs will work fine as v1.3 DTD. The main change is the addition of a `@class` attribute to every element, which enables the "extra-css" section in the skinconf to be put to good use.

2.3. Changes between document-v11 and document-v12

doc-v12 enhances doc-v11 by relaxing various restrictions that were found to be unnecessary.

- Links ((link|jump|fork) and inline elements (br|img|icon|acronym) are allowed inside title.
- Paragraphs (p|source|note|warning|fixme), table and figure|anchor are allowed inside li.
- Paragraphs (p|source|note|warning|fixme), lists (ol|ul|dl), table, figure|anchor are allowed inside definition lists (dd) and tables (td and dh).
- Inline content (strong|em|code|sub|sup|br|img|icon|acronym|link|jump|fork) is allowed in strong and em.

This is a legal notice, so it is **important**.