1. NewbieDoc writer instructions

Jesse Goerz

Revision History

Revision v0.1 23 March 2001 Revised by: jwg
Initial release.

This document is intended to help new doc writers publish sgml documents to the newbieDoc sourceforge project. In an effort to ease the burden on administrators we ask that you follow the conventions in this document. Doing so will allow your documents to be incorporated into the newbieDoc archive much more rapidly.

1.1. The Lay of the land

What's in an sgml document? An sgml document in it's most basic form looks terribly similar to an html document. In fact, html is derived from sgml. Below is an example of what a sgml document looks like.

```
<!DOCTYPE sect1 PUBLIC "-//OASIS//DTD DocBook V3.1//EN">

<sect1 id="hello world"><title>Hello world</title>

   <para>
   Hello world!
   </para>

   <sect2 id="goodbye_world"><title>Goodbye world</title>
      <para>
      Goodbye world.
      </para>
   </sect2>

   <sect2 id="goodbye_world"><title>Goodbye again</title>
      <para>
      Goodbye again.
      </para>
   </sect2>

   <para>
   Hello again!
   </para>

</sect1>
```

To illustrate a common error I have intentionally made several mistakes in the preceding example. All of the *id* parameters are of an incorrect format and I used the same "id" name twice. If I tried to "build" this sgml file with an sgml parser it would return an error like this:

```
jesse@storm:~/test$ sgmltools -b html simple.sgml
/usr/bin/jade:<OSFD>0:2:11:E: value of attribute "ID" must be a single token
/usr/bin/jade:<OSFD>0:8:15:E: character "_" is not allowed in the value of attribute "ID'
/usr/bin/jade:<OSFD>0:15:23:E: ID "GOODBYE_WORLD" already defined
/usr/bin/jade:<OSFD>0:9:15: ID "GOODBYE_WORLD" first defined here
```

Which just simply means you can't have spaces or underscores in the name of your "id" attribute, and you need to be careful not to duplicate id names in your document. This may seem trivial right now, but it will be important later so remember this example.

What's so special about our sgml documents? Well, nothing really, but we do organize them in a specific way to try and make keeping them up to date as easy as possible. First of all, we use three types of documents here at NewbieDoc; Books, Chapters, and like the one you just saw, sections. We keep them in separate files so they are easier to maintain by individual writers. Then we use entity declarations to link them all together. Don't worry if you don't know what those are we will cover that a little later.

1.2. The hierarchy

Books, Chapters, and Sections. Our documents are organized in a simple fashion. Docbook sgml facilitates this with the use of the book paradigm. If you think about it, it makes sense. A book has several chapters, and each chapter has many pages (in our case we use sections rather than pages.) Our entire archive is represented by the book. The main file which holds the book together is `index.sgml` located in the root of `/newbiedoc`. Think of this as the cover and table of contents of our book.

Next we have many chapters. The chapters of our book are represented by different types of categories. For instance, because we are writing about Linux we may have a chapter on system administration, networking, text editors, and/or utilities. Each of these chapters would itself contain many sections. For example, the text editors chapter could contain a section on vi, joe, emacs, and/or ed. Putting all these sections in the text editors chapter makes them easy to organize and easy to change if we need to.

Entities: The linkers. Entity is just a fancy name which docbook uses to mean a cross between a variable and a sym-link. Here is what a chapter looks like with some entity declarations:

```
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook V3.1//EN"
   [
<!ENTITY sourceforge-guide SYSTEM "metadoc/sourceforge-guide.sgml">
<!ENTITY template SYSTEM "metadoc/template.sgml">
   ]>

<chapter id="gendoc"><title>General Documentation</title>

<!-- Section 1 sourceforge-guide -->
  &sourceforge-guide;

<!-- Section 2 document template -->
  &template;

</chapter>
```

Since I didn't go into to much detail earlier, I'm going to cover a little bit more then just the entities. On the first line you have what is known as the DTD or Document Type Declaration. This is just a fancy way of saying what type of document it is. In this case it's a chapter. In the earlier example we looked at a section. Take a minute and compare the two. One thing to take note of is that not only are their names different but the section DTD ends with a ">" right after `//EN"`. The chapter DTD, however, does not end until after all the entities have been declared. Also note that the entities are all inside their own little container; they have a starting bracket and an ending bracket, and only then does the DTD terminate with a ">".

Entities declarations take the form:

```
<!ENTITY name-of-entity SYSTEM "relative/path/to/file.sgml">
```

Once the entities are declared in the DTD, think of them as variables holding a pointer or URL link to the file included within them. Now that they are declared you can import the files they point to by invoking them as an entity. This is done by preceding the name of the entity with the "&" (ampersand) and ending it with a ";" semi-colon. Once again, look at the chapter file above to see an example.

It is important that you *distinguish the difference* between *entity declarations* and entities. Entity declarations hold the directions to get to a file, but they do absolutely nothing until they are told to. Entities invoked within the document like " &sourceforge-guide; " in the preceding example tell their declarations to import the file into the document right at that point.

Tying it all together. The reason you need to understand entities is because it will help prevent you from making simple errors which might cause the book build to fail. The book build is when the administrators build all the sections into their appropriate chapters, and all the chapters into the book. Most of us will only need to edit the book and chapter files when we are adding a new document. After that you can pretty much leave them alone unless you change the name of your sgml document file. I'll cover what you need to do to add or update a new document in a later section.

1.3. NewbieDoc document conventions

Directory/File names. Directory names which fall under `/newbiedoc` should be pretty self-explanatory. Most directory names will be the same as the chapter names. For example, the `system-admin` directory contains all the section documents which cover subjects related to system administration. In the root of `/newbiedoc` there would be a `system-admin.sgml` which would be a chapter document with entities pointing at all the section documents within the `system-admin` directory. Here's what it would look like in your working directory:

Remember `index.sgml` in the root of `/newbiedoc`? That is the cover and table of contents of the book; it is what holds it all together. The administrators use that file to do the book build. Contained inside `index.sgml` are all the entity declarations.

```
jesse@storm:~/cvs/newbiedoc$ ls -l
total 84
drwxr-sr-x    2 jesse    jesse        4096 Mar 23 19:15 general-doc
-rw-r--r--    1 jesse    jesse           0 Mar 23 19:17 general-doc.sgml
-rw-r--r--    1 jesse    jesse        4110 Mar  1 16:56 index.sgml
drwxr-sr-x    2 jesse    jesse        4096 Mar 23 19:14 system-admin
-rw-r--r--    1 jesse    jesse           0 Mar 23 19:16 system-admin.sgml
drwxr-sr-x    2 jesse    jesse        4096 Mar 23 19:16 text-editors
-rw-r--r--    1 jesse    jesse           0 Mar 23 19:17 text-editors.sgml
drwxr-sr-x    2 jesse    jesse        4096 Mar 23 19:16 utils
-rw-r--r--    1 jesse    jesse           0 Mar 23 19:17 utils.sgml

===================Directory listings below here ============================

jesse@storm:~/cvs/newbiedoc$ ls -l general-doc
total 36
-rw-r--r--    1 jesse    jesse       13126 Mar  1 16:55 sourceforge-guide.sgml
-rw-r--r--    1 jesse    jesse       19285 Mar  1 16:56 template.sgml

jesse@storm:~/cvs/newbiedoc$ ls -l system-admin
total 24
-rw-r--r--    1 jesse    jesse       23566 Mar  1 16:56 runlevels-intro.sgml

jesse@storm:~/cvs/newbiedoc$ ls -l text-editors
total 32
-rw-r--r--    1 jesse    jesse       11139 Mar  1 16:56 joe.sgml
-rw-r--r--    1 jesse    jesse       18598 Mar  1 16:56 vi.sgml

jesse@storm:~/newbiedoc$ ls -l utils
total 8
-rw-r--r--    1 jesse    jesse        6390 Mar  1 16:56 grep.sgml
```

Entity names. Entity names should be the same as the file they point to in their declaration. When two file names in different subdirectories happen to have the same name the subdirectory name will be included in the entity name. For example:

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook V3.1//EN"

[
```

```
<!-- These entities import the chapter files.  -->
<!ENTITY general-doc SYSTEM "general-doc.sgml">
<!ENTITY system-admin SYSTEM "system-admin.sgml">
<!ENTITY text-editors SYSTEM "text-editors.sgml">
<!ENTITY utils.sgml SYSTEM "utils.sgml">

   <!-- These entities import the section files for   -->
   <!-- General Documentation                         -->
<!ENTITY sourceforge-guide SYSTEM "general-doc/sourceforge-guide.sgml">
<!ENTITY template SYSTEM "general-doc/template.sgml">

   <!-- These entities import the section files for   -->
   <!-- System Administration                         -->
<!ENTITY runlevels-intro SYSTEM "runlevels-intro.sgml">

   <!-- These entities import the section files for   -->
   <!-- Text Editors                                  -->
<!ENTITY text-editors-joe SYSTEM "text-editors/joe.sgml">
<!ENTITY text-editors-vi SYSTEM "text-editors/vi.sgml">

   <!-- These entities import the section files for   -->
   <!-- Utilities                                     -->
<!ENTITY utils-grep SYSTEM "utils/grep.sgml">
  ]>

<book id="index" lang="en">
```

Even though `vi.sgml` and `joe.sgml`'s entity names are different (vi and joe respectively) I chose to include their subdirectory names in their entity names as a precaution. Both of these editors are very commonly brought up in howto's and usually have their own section with a link. The name of the link is usually vi or joe. To avoid confusion and a possible conflict we use their subdirectory name along with their filename as the entity name.

URL/link names. The best way to avoid duplicate links is to append the filename to the link you are creating. This way when the section documents are imported into the book, the administrators won't have to deal with changing 5 or 6 section documents who all have url links to "intro". As an example we will fix the first section document I showed you at the beginning of this tutorial. We will assume the file is called `howdy.sgml`.

```
<!DOCTYPE sect1 PUBLIC "-//OASIS//DTD DocBook V3.1//EN">

<sect1 id="intro-howdy"><title>Hello world introduction</title>

   <para>
   Hello world!
   </para>

*this file left incomplete intentionally*
```

As you can see, now that I have appended "-howdy" to the end of the link, it is highly unlikely that when the administrators do a book build and all the section documents are imported into one file to be processed that the links will have conflicts.

1.4. Basic templates

Sections. Please notice that these templates all include a GNU Free Document License at the top. This is required so that each individual section is covered by our license. If you use your own templates you must include this license at the top of your sgml file. Please make sure it is commented out. If this license is unacceptable to you, then you should be publishing documents elsewhere.

```
<!--
Copyright (c)  2001  NewbieDoc project at
http://sourceforge.net/projects/newbiedoc
Permission is granted to copy, distribute and/or modify this
document under the terms of the GNU Free Documentation License,
Version 1.1 or any later version published by the Free Software
Foundation; with no Invariant Sections, with no Front-Cover
Texts, and with no Back-Cover Texts. A copy of the license can
be found at http://www.fsf.org/copyleft/fdl.html.
-->

<!-- want to build this document? Uncomment the line         -->
<!-- directly below. Make sure you comment it back out       -->
<!-- before you commit it back to cvs.                       -->

<!--
<!DOCTYPE sect1 PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
-->
<sect1 id="name-of-document"><title>Name of document</title>

   <sect2 id="intro-nameoffile"><title>Introduction</title>
   <para>
   some filler
   </para>
   </sect2>

   </sect2 id="changelog-nameoffile"><title>Changelog</title>
   <para>
   some filler
   </para>
   </sect2>


</sect1>
```

Chapters.

```
<!--
Copyright (c)  2001  NewbieDoc project at
http://sourceforge.net/projects/newbiedoc
Permission is granted to copy, distribute and/or modify this
document under the terms of the GNU Free Documentation License,
Version 1.1 or any later version published by the Free Software
```

```
Foundation; with no Invariant Sections, with no Front-Cover
Texts, and with no Back-Cover Texts. A copy of the license can
be found at http://www.fsf.org/copyleft/fdl.html.
-->

<!-- Uncomment the declaration directly below this when you  -->
<!-- want to build this entire chapter without building      -->
<!-- the entire book.  Make sure you comment it back out     -->
<!-- before you commit it back to cvs.                       -->
<!-- Don't forget to add an entity declaration to index.sgml -->

<!--
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook V3.1//EN"
   [
<!ENTITY sample SYSTEM "put_your_doc_here.sgml">
<!ENTITY sample1 SYSTEM "put_your_doc_here.sgml">
<!ENTITY sample2 SYSTEM "put_your_doc_here.sgml">
   ]>
-->

<chapter id="name-of-chapter"><title>Name of Chapter</title>

<!-- Section 1 sample-->
  &sample;

<!-- Section 2 sample1 -->
  &sample1;



</chapter>
```

1.5. Adding new documents

New documents should be added to the appropriate chapter subdirectory. Using the previous example, if I added a new section document for emacs it should go into the `text-editors` directory. I would then add an entity in the text-editors.sgml file pointing to this new file, and an entity declaration in the index.sgml file. Before committing to cvs it is a good idea to "build" the chapter with your sgml parser and make sure everything builds without errors.

Here's a quick checklist:

- Compose your new section document. Include the GNU Free Document License at the top of the sgml file. Run your sgml parser on it and confirm there are no errors.
- Add an entity and entity declaration to the appropriate chapter file. Run your sgml parser on the entire chapter and make sure it and your new section doc build without errors.
- Add an entity declaration to the index.sgml file.

- Comment out the DTD declaration at the top of your section document and commit it to cvs. If you ran your sgml parser on the entire chapter comment out the entity declarations (only the declaration, leave the entity in there) in the chapter file before you commit it to cvs.

1.6. Updating your documents

The only thing different about updating your documents versus adding them as new documents is you have to trace down and update all the entities and entity delcarations if you happened to change a filename.