AN XML DTD FOR PROJECT GUTENBERG

Cynthia L. Blue

A Master's project submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina

April, 2001

Approved by:

_____

Advisor

Abstract

Cynthia Blue. An XML DTD for Project Gutenberg. A Master's project for the Master's of Science in Information Science degree. April, 2001. 76 pages. Advisor: Gregory B. Newby.

Project Gutenberg is an electronic collection of documents and literature, the majority of which exist in ASCII format. While the ASCII format has been an almost universally accessible format since the Project started in 1971, the possibilities and advantages of marking up the texts with the Extensible Markup Language (XML) are compelling. Related efforts are detailed and analyzed for viability with the Gutenberg texts. This project presents a direction for the future of this effort and a DTD suitable for the collection. The prepared DTD provides the schema against which 5 test documents are marked up with XML. A tutorial based on my experiences marking up the text and an index of the available elements are included.

Headings:

XML (Document Markup Language)

Document Type Definitions (DTDs)

Schemas

An XML DTD for Project Gutenberg

Project Gutenberg is an online collection of documents and literature that is freely available to anyone with Internet access around the world. Michael Hart started this project in 1971 when he received an account and one hundred million dollars worth of computer time at the University of Illinois Materials Research Lab. He felt that "the greatest value created by computers would not be computing, but would be the storage, retrieval, and searching of what was stored in our libraries" (Project Gutenberg, 1992). Upon this windfall, he decided to use his account to convert literature to plain ASCII text, in order to create a universally available electronic repository of literature and documents. The first document on the network was the "Declaration of Independence," and texts such as the King James Bible and Edgar Allen Poe poetry followed. The collection's only restrictions are that the texts would have a large audience, and are in the public domain, or otherwise have authorization to appear on the site. Throughout the remainder of this paper, the word "text" is used to summarize the whole host of genres contained within the Gutenberg collection.

In 1971, not nearly as many electronic formats were available as exist currently. The reason that the ASCII format was chosen was due in major part to the fact that 99% of the world's computers can read these characters. The Gutenberg philosophy of a universally available electronic document was supported by this format, and still is today. While many have wondered about the viability of this format, as compared to others, the ASCII format of text has been the only that could meet the following criteria:

- Ease of use, as was important to the philosophy of the project;
- As universally available as possible;
- Cost effective;
- Viable as operating systems, hardware and software change.

Today, the Project maintains over 4,000 "plain vanilla ASCII" texts that do not contain additional markup. In regards to this issue, the Project website reads:

> Thus any complaints about how we do italics, bold, and the underscoring, or whether we should use this or that markup formula are sent back with encouragement to do it any ways any person wants it, and with the basic work already done, with our compliments… We need to have e-texts in files a Plain Vanilla search/reader program can deal with; this is not to say there should never be any markup…just those forms of markup should be easily convertible into regular, Plain Vanilla ASCII files so their utility does not expire when programs to use them are no longer with us (Project Gutenberg, 1992).

The languages and applications that are currently available to process texts are changing rapidly. If the project were to add another format, finding a stable markup language would be key. Project Gutenberg cannot commit to a language that would require one to go through and modify each text as the markup language develops. In recent years, how often have we had problems opening documents that are in previous or latter version of the software that we are using, or that was developed in software that no longer exists? We have become accustomed to updating our documents to the latest version of software that we are using, but the ASCII texts have been functional and accessible for 30 years.

Another very important factor to consider when attempting to change, reformat or markup the Gutenberg texts in any way is that the size of the collection will prohibit volunteers from effectively maintaining the texts. Certainly, scripts can be written that

could easily modify the texts, but the size and variety of the collection would prohibit any thorough quality assurance efforts. A plain ASCII version of the text should always be available, but the Project would be prematurely limiting options if new standards for markup were never investigated.

In 1971, providing free, universally available texts was a revolutionary step as disk space and processing speed were limited. However, current times have brought drives with gigabytes of storage space and extremely fast processors. Some markup languages can chunk the texts into separate files that will take up less storage space. Most importantly, though, the users' expectations of electronic texts have changed; expectations of appearance, format and usability have increased. Markup languages can provide an efficient way to meet these expectations. Of all of the markup languages and variations upon the languages that have appeared in the last 30 years, one in particular has received a lot of attention in the past few.

XML Introduction

In the past few years, the attainable glories that the Extensible Markup Language (XML) could provide to web applications have been proclaimed, questioned and tested. While this introduction will not provide a detailed overview to XML, its functions and possibilities must be introduced.

XML's roots lie in the Standard Generalized Markup Language (SGML). SGML was developed as means for indicating the meaning and structure of documents, but was so complex that widespread usage was never realized. The Hypertext Markup Language (HTML), used for formatting purposes, proved to be a small but functional enough subset of SGML to encourage greater usage. However, as the number of web developers

proliferated, the use of the HTML tags was pushed beyond stylistic purposes. The myriad possibilities for Internet applications became clearer and included such applications as data transfer, document management, information sharing and e-commerce. The World Wide Web Consortium (W3C) started a working group for a new subset of SGML, XML, in January 1997. The group "proposed a markup language that could work in concert with existing Web technologies, using some of the tools developed for use with HTML, while moving forward with more manageable techniques" (St. Laurent, 1999, p.11).

The philosophy of XML is based on a few fundamental principles (Usdin & Graham, 1998).

1. **Separation of Content from Format:** The role that a piece of information plays should be distinguishable from the information's appearance. Information's use, role, or nature in a particular application should be identified. For example, "knowing that a phrase is in italic is useful; knowing that it is the title of a subsection of a paper is more useful; and knowing that it is a genus and species name is potentially more useful still" (p.126).
2. **Hierarchical Data Structures:** In XML, the data is assumed to be hierarchical; a piece of information may contain other pieces of information. For example, a book contains several chapters, each of which contains sections. Each section may have a heading, paragraphs and subsections, which also contain a heading and paragraphs.

3. **Embedded Tags:** XML documents consist of tags that identify where the data structures begin and end. Tags can have attributes that provide additional information about the data within the tags.
4. **User-Definable Structures:** As mentioned above, XML is a tool, and it defines a method of customized tag creation while still providing flexibility and extensibility by not providing a standard tag set.

XML and Project Gutenberg

The Gutenberg texts could benefit from being marked up in XML. Particularly:

- ASCII presentation is bland and readers' expect more. With XML, conversion of the text for different formats is simple.
- ASCII only supports the English character set; XML's default is 8-bit Unicode, which allows for alternate character sets.
- The current documents have unstructured content which presents a major drawback for searching; XML can describe the structure and content of the texts (Boumphrey, 2000a).
- Texts could easily be converted back to plain ASCII text.
- Formatting can be applied to the texts with the use of Cascading or Extensible style sheets.

From conception, the purpose of this project was to design a Document Type Definition (DTD) for Project Gutenberg. However, very quickly upon beginning research on the topic, many other viable options surfaced. Formatting, downloading, searching, or otherwise processing the texts are the fundamental reasons for marking up the texts, and often drive the DTD development process. However, no true standard has

emerged from among the various electronic publishing efforts. "There are many other scattered efforts, and there is indeed a great need for a source to co-ordinate and centralize all these efforts, so that they do become truly available to everyone" (Boumphrey, 2000a).

Many factors that have been presented in the previous sections introducing Project Gutenberg and XML are worth repeating here. Before the various available DTDs are detailed, the fundamental philosophies of Project Gutenberg and the value that XML can add to the texts should be kept in mind.

1. Any DTD used for Project Gutenberg should be free and publicly available, a concept that is at the center of the Project Gutenberg philosophy.
2. In order for a majority, and hopefully all, of the documents to be marked up by volunteers, the DTD must be intuitive and easy to learn.
3. The DTD must be flexible enough to accommodate all different types of tests, from government documents to poems. "The aim is to…be suitable for: books, poetry, plays, saga's, diaries, compendiums, letters, mixed content, atlases, encyclopedias, dictionaries, historic documents, scientific documents and parallel translations" (Boumphrey, 2000b).
4. The processes that will most likely be applied to the texts are formatting, searching, downloading; DTDs designed for more complex application processing may be unnecessarily complex.

The various electronic text markup efforts provide varying advantages and disadvantages for the Gutenberg collection. Each of these efforts is presented below, and must be considered with all of these issues in mind.

Related Efforts and Approaches

Open eBook Initiative

The Open eBook (OeB) Initiative is an organization concerned with bringing together various markup standards in the eBook and epublishing worlds. The OeB released the final 1.0 version of the Open eBook Publication Structure specification in September of 1999. It defines the format for electronic content and a standard for representing the content of electronic books. Specifically, the specification intends to: (a) give eBook content and tool providers common guidelines to ensure accessibility and consistent presentation of electronic content over various eBook platforms, (b) reflect established content format standards, (c) provide the suppliers of electronic-book content a format to use for supplying to multiple reading systems.

The specification is based on the assertion that electronic-book technology cannot achieve widespread success in commercial markets unless reading systems much have convenient access to a large quantity and variety of texts (Open eBook Forum, 2000).

A noteworthy point on the eBook standard is that it uses a combination of HTML, XHTML and XML. While HTML is the markup used for formatting and appearance, and XHTML is a structural language that describes the structure but not the content, this combination of markup languages may not be desirable for Project Gutenberg's movement towards a straight XML markup. "The consortium's goal is to bring the Open eBook standard as close to XML as possible. It will, however, take a few years to reach that point, allowing time for hardware designs to catch up. Current e-book designs use processors that are not powerful enough to render XML code" (Spooner, 1999).

Although the OeB is not pure XML, it is based on the general belief that its flexibility and simplicity will support the lifespan of electronic documents, and support compatibility and interoperability across systems. "[P]articipants…are concerned mostly with an interoperable data specification …[to] promote the rigorous separation of structure and rendering semantics, …to reconcile supporting innovation with maintaining interoperability, and establish a foundation for supporting internationalization." (Cover, 2001). Over 100 participants have collaborated on these standards including such well-known and influential organizations as Microsoft and SoftBook Press (DeRose & Renear, 2000).

This specification defines two DTDs, the package DTD and the basic OeB document DTD. The package provides the base of the publication which reading systems would use to find and organize the text's components. The basic OeB document DTD maps the HTML subset described in this specification (Open eBook Forum, 1999).

Project Gutenberg would certainly benefit from the knowledge imparted by the OeB efforts. However, since many of the biggest commercial players in the eBook and epublishing markets have been involved with its development, one needs to consider the role of the Gutenberg texts in the eBook world. The Gutenberg texts would greatly benefit from some form of markup for flexibility of format, appearance and processing, but they were never intended to compete with the publishers and other content providers. Also, the OeB is concerned with hardware developers, whereas Project Gutenberg may not take the needs of hardware developers into consideration. Furthermore, these standards are currently fraught with contention and debate, and may continue to change for quite sometime into the future, contrary to the previously established requirement of a

stable DTD for Project Gutenberg. Thus, some lessons can be learned from the OeB, and perhaps incorporated into the DTD, but the OeB package DTD cannot be considered a complete solution for the Gutenberg markup efforts.

Project Perseus

Project Perseus at Tufts University is a Digital Library that includes various SGML and XML tools for document management and analysis. The tools were originally developed for Ancient Greek, but were extended for use with Latin and Italian. Their efforts include storage, morphological and lexical analysis, metadata and cataloguing efforts, and display of texts (Mahoney, Rydberg-Cox, Smith and Wulfman, 2000). The Project is notable and worth watching, and some interesting lessons can be extracted from their efforts. Currently, they are trying to generalize their tools for use with other languages and projects. The system developed could be packaged and offered to similar projects as open source in the near future.

Upon first researching the Perseus efforts, it seems as though many of the same concerns and difficulties of marking up the Project Gutenberg texts has also been considered with their efforts. "One of the greatest challenges in building and maintaining a large, heterogeneous [Digital Library] is the necessity of managing documents with widely varying encoding and markup practices…varying DTDs…" (Smith, Mahoney, Rydberg-Cox, 2000). Texts in the collection vary in structure and use diverse DTDs. The most common DTD used for Project Perseus texts is based on TEI, but the abstract meanings of the elements of all DTDs used are mapped to one structure. This system attempts "to extract structural and descriptive metadata from these documents and deliver document fragments on demand; and to support other tools that analyze linguistic and

conceptual features and manage document layout…" (Smith et al, 2000). While Project Perseus is a very flexible and modular document management system, it is way too complex for Project Gutenberg.

DocBook

"Because DocBook is a large and robust DTD, and because its main structures correspond to the general notion of what constitutes a 'book,' DocBook has been adopted by a large and growing community of authors writing books of all kinds" (Cover, 2000).

DocBook is an SGML DTD that was designed for marking up books about computer hardware, software, and other technical documentation. It was started by Norman Walsh in 1991, but was recently turned over to a technical committee of the Organization for the Advancement of Structured Information Standards (OASIS) for maintenance and development (Walsh & Muellner, 1999). While the official DocBook distribution is an SGML DTD, an XML DTD based upon DocBook version 3.0 has been under development for some time.

While DocBook is perhaps the most fleshed out and widely used DTD, the simplified version alone was 26 printed pages on 8½" by 11" paper in `10-point courier font`. While this one DTD may accommodate several different types of texts in Project Gutenberg, the lengthy and complex detail that lends to the DTDs flexibility may be a hindrance to volunteers' support. Even to people experienced with XML, the notation is difficult to learn, and finding the elements that would be needed to markup a simple piece of literature from the Gutenberg collection proves complicated. "DocBook is not really suitable for working with literature, though it is (obviously) fantastic for technical documentation" (Meggison, 2000). Furthermore, Norman Walsh owns the

copyright to the DTD, which is unlike most items in the collection that are in the public domain.

HTML Writers Guild

The HTML Writer's Guild (HWG) is a not-for-profit educational organization of self-proclaimed HTML professionals that support the use of good quality markup in web pages. In a volunteer effort to help markup the Project Gutenberg texts, members worked to design a set of XML DTDs for the collection, and to encourage their members and others to participate in the markup. While it seems that activity on this effort has waned since the Spring of 2000, over 50 texts were marked up as of March 2000 in response to their activities. "There are numerous DTDs that can be used…we are encouraging everyone to use either XHTML, or one of our modular DTDs or TEI…any DTD you want, however it must either be widely available to the public…" (HTML Writers Guild, 2001).

Frank Boumphrey, now President of the HWG, was very active in this effort; he wrote the DTDs and all of the online documentation for this project. In an online discussion forum about the differences between the various standards efforts and their own, Mr. Boumphrey stated:

> We have an open mind about eBook. Indeed we have an open mind about all DTDs. We have people marking up documents against DocBook, TEI and XHTML. If any one wants to use eBook to mark up books that's fine by us, all we ask is that the DTD used is freely available.
>
> My own reservations about eBook are that it appears to mix css with XHTML for a styled markup, and I feel that markup of historic documents should be entirely semantic plus structure. So if that is what the marker wants to do I would prefer XHTML plus a style sheet, but that is purely a personal opinion, officially the marker can use eBook….

> In fact 90% of the books marked up so far have used one of our own DTDs, which are designed to be both descriptive and easy to use (2000c).

The Text Encoding Initiative

The Text Encoding Initiative (TEI) was established in 1987 in the hopes of reducing the number of existing encoding practices and encouraging the sharing of electronic texts. The TEI SGML DTD is large, flexible, and very widely used. Members of the research and academic community with interests in the humanities computing community, worked to develop a common method for encoding textual structures. Under the auspices of an international cooperative project and with support from the Association for Computers and the Humanities, the Association for Computational Linguistics, and the Association for Literary and Linguistic Computing, the TEI DTD was developed.

Participants felt that their scheme could easily be extended to other text encoding efforts, and considered TEI to be able to meet the markup requirements of myriad disciplines. “Thus, the TEI became the only systematized attempt to develop a fully general text encoding model and set of encoding conventions based upon it, suitable for processing and analysis of any type of text, in any language, and intended to serve the increasing range of existing (and potential) applications and use” (University of Virginia Library, “…: Note”).

Specific objectives of TEI include the creation of a standard format for data interchange and guidance for marking up texts in this format, providing a scheme that would work for any feature studied by researchers, while remaining application independent (University of Virginia Library, “…:Underlying principles”).

Several efforts to manage electronic texts and digital libraries include texts marked up with the TEI DTD. Oxford University and the University of Virginia both

maintain large collections of electronic texts marked up with the TEI SGML DTD (Boumphrey, 2000a). Simplified versions of the full DTD, also known as TEI Lite, seem to be popular, and are intended to be easier to learn and use. An XML version of the DTD is also under development.

Because of its roots in the humanistic research community, the TEI scheme would likely be the closest option to a complete standard that would work for marking up the Project Gutenberg texts. The Initiative is faithful to their goal of providing high levels of clarity, flexibility, and extensibility; however, this DTD is also very complex and difficult to learn, and would certainly require that very technically proficient volunteers markup up the texts.

Summary and Rationale

With the groundwork for Project Gutenberg and XML laid, and the various electronic text efforts and DTDs presented, the remainder of this paper details the decisions that guided my project. While keeping in mind that the fundamental goals of marking up the Gutenberg texts are to maintain an electronic version of texts in the public domain that are as universally available as possible while maintaining an easy to use format at a low cost, the means are clear. While DocBook is one of the oldest, most widely used DTDs available, its origin in technical documentation becomes quickly apparent in its difficulty to read and understand. TEI is also fairly difficult to learn, although it comes closer to the needs of Project Gutenberg in that it was developed around the humanities. Project Perseus was developed for Ancient Greek texts, and may perhaps be an effective extension of the TEI Lite DTD, but its concentration on morphological analysis and presenting lexica make it a much more powerful tool than

necessary for Project Gutenberg's needs. The HTML Writer's Guild has put together some very user-friendly yet comprehensive DTDs specifically designed for the Gutenberg texts, and thus seems the right direction for this project. Their DTDs, though, could be improved upon to extend their capabilities, simplify use, and facilitate management of the texts. Their coverage of the parts of a book, poem, and play seem very comprehensive upon first examination, and were in part derived from The Chicago Manual of Style. Adopting this structure would serve as a great advantage to this effort by eliminating that initial process. For these reasons, the remainder of this paper will detail the current DTDs, how and where I feel they can be improved upon, and a DTD with my suggested changes.

A few issues regarding the logistics and direction of the DTD should be considered before proceeding with the details. First, the HWG efforts allow and encourage markers to use any DTD they feel comfortable with; the advantages of allowing this may or may not outweigh the potential processing problems. While allowing the use of multiple DTDs may encourage more volunteers to participate in marking up the document, this may hinder effective searching of the collection. "A programmer wishing to extract all of the book titles mentioned in a collection of documents marked up in varying DTDs may have to look for <cit> in some documents and <title> in others, whose DTD might use <cit> to mean a piece of quoted text" (Smith et al, 2000). If the primary purpose of marking up the Gutenberg collection is purely aesthetic, then use of varying DTDs is acceptable. However, if these types of processing issues would burden future applications, than only one DTD should be supported. The

managers of Project Gutenberg should ultimately decide the guidelines for marking up the documents with varying DTDs based on their goals.

Another outstanding decision is concerned with the documents that have already been markup up by the HWG volunteers. For example, should an element name change in the new DTD to a more intuitive name, should it matter that these documents will not work with the new DTD? I decided not to concern myself with the texts that have already been markup up, partially because the marked up documents number only about 50, a seemingly manageable number to revise, and because the goal of this project is to create a DTD to meet the needs of the Project, not to accommodate what has already been done.

While this DTD will not conform other publishing standards, it should still meet the basic requirements of the XML Specification from the W3C, in order to potentially increase the lifespan of Gutenberg XML documents, including:

- all documents are well formed;
- documents have the correct XML declaration;
- encoded in UTF-8 or UTF-16;
- empty elements uses only the empty element syntax with white space before the trailing slash;
- all element and attribute names must be in lower case.

Finally, if one single DTD is to be successful, the DTD must be flexible enough to accommodate almost any possible document structure. Few constraints should be placed on the order of elements, or the requirements on them. Furthermore, some 1,700 volunteers (Miller, 2001) that currently participate in the maintenance of the project may

be called upon to help to markup the documents. While someone may feasibly create scripts that will markup the documents automatically, the DTD should still be as intuitive and flexible as possible for volunteers to use.

Methodology

Top-level Structure

The HTML Writer's Guild DTDs are four comprehensive, but individual DTDs: gutbook1.dtd for books, gutplay1.dtd for plays, gutpoems1.dtd for poems, and gutbkplay1.dtd for books with pieces of plays in them (HTML Writer's Guild, 2001b). Each of these four DTDs contains only the top-level DTD structure for the Gutenberg and contains a reference to another DTD that contains the elements for the actual document content. In other words, the actual "gutplay" DTD only consists of elements for the metadata, and refers to another DTD, "playfrag" as an entity with `<!ENTITY % playfrag SYSTEM "playfrag.dtd"> %playfrag;`.

The benefit to structuring a DTD in this manner is not evident. My assumption is that the purpose of this scheme was to separate the actual text from the Gutenberg information; however, this is really not necessary with the processing capabilities intrinsic to the XML language. This structure requires that the volunteers that are marking up the documents understand the eight separate DTDs and how they work together and contrast. Consequently, a volunteer that has marked up several documents might remember an element or attribute that was available within the "titlepage" section of one DTD and attempt to use it when working with another DTD that does not use this element, or use it in the same way. By combining and sharing all of the available elements, the initial learning curve for a volunteer marking up documents is steeper, but then they only have to learn and refer to one DTD.

Upon reviewing the four DTDs, the amount of overlap between them was immediately evident. Clearly, only three major elements are available: book, play, and poem. While a fourth DTD covers the combination of a book with a play fragment in it, I decided not to consider it as all of the elements that a book might need are available in the book DTD, and all of the elements of a play are available in the play DTD. Furthermore, these DTDs could not accommodate the government documents and speeches that exist in the Gutenberg collection. Also, an overlap appears in the book DTD, which contains a reference to a poem element that was only slightly different than the complete, independent poem DTD. The complete poem DTD provides for more elements than the poem element in the book DTD; all poem elements should be available to poems that are within a book structure.

A new DTD should not be created for each new subset of elements. For example, perhaps the collection contains a government document that contains elements that only partially exist in the book DTD. Instead of pulling out all of the elements that are the same and creating a new DTD with this new subset of elements, these new elements should simply be added to the existing DTD. They are then also immediately available to any of the other documents in the collection. Not only does combining these top-level elements into one document simplify our DTD, but all elements and attributes are now available to any document that needs to use them. Furthermore, maintaining several DTDs may threaten elemental consistency throughout the collection and make processing more difficult.

Thus, I chose to create one DTD, extend it for use with miscellaneous documents, and extract all of the metadata into external entities. Since the book DTD seemed to be

the most comprehensive of the three DTDs, it was used as the base into which the remaining DTDs were compiled. Any elements from the poem or play DTD that did not exist in the book DTD were added to the book DTD.

I provide the original authors complete credit for their work, and provide my changes and suggestions strictly as an offering of opinion and not a presentation of their own research and efforts as my own. However, I would hope that this DTD remains public and freely available to anyone who would like to extend or improve it.

Modifying the Parts – Elements, Attributes and Entities

The elements of the three DTDs -- book, poem, and play -- overlapped tremendously. In Appendix A, Figure 1, the top-level structure of each of the HWG's poem, book and play DTDs are diagrammed as organizational charts. As demonstrated, most of the top-level elements are the same between the three. Many of the sub-elements and leaf elements (elements that have no children), are identical between these DTDs. More accurately:

- 13 elements were the same between all three of these DTDs;
- 44 elements, in addition to the 13 above, were the same for the book and play DTDs;
- Only 17 play-specific elements were added to book;
- 16 elements existed in book that were not available in poem or play;
- The original poem element in book was deleted.

By defining a group of DTD-wide attributes, they can easily be added to any element with a simple reference. "DTDs that include a significant set of child elements that can be used in multiple parent element can be simplified with parameter entities

listing the elements. The parser should parse the parameter entity and add its markup to the element content declaration" (St. Laurent, 1999, p.136). I renamed the DTD-wide attributes "dtdattribs," as that seemed to me to be more intuitive to new users than the current "stdatts" name used.

The inline.class and block.class declarations were not included in the poem DTD, but were available in book and play, thus remain in the combined DTD. Block and inline entities are used for parts of text that can appear anywhere, and are mainly defined as inline or block based upon their general appearance within the text. Inline entities are ones that appear within a line of text, but can appear anywhere within the text and therefore do not have specific requirements for use within the elements. Block entities are used in a similar manner, but appear within a separate paragraph or aesthetic chuck of a text. An example of an inline element is a date, and a block entity might be a blockquote or a table. Any references similar to `<!ELEMENT acknowledge (#PCDATA | %inline.class;)*>` declares that the acknowledge element can contain PCDATA and any of the elements declared in the inline.class entity.

In addition to the inline and block entities, entities have another very important function in this DTD. XML has the capability to chunk documents into sections and present them together seamlessly in the browser window by using entity and general parameters. "General entities are simple and make many complex and annoying tasks very simple, especially when it comes to filling in boilerplate text….Creating entities this way is useful for repetitive information that is prone to change during the lifetime of the document" (St. Laurent, 1999, p.153). This use of XML could benefit the Gutenberg texts greatly by pulling out all of the Gutenberg-related metadata that is the same across

all of the texts into a separate entity or entities, that would be stored in separate files and referenced or pulled into the XML document. Similar to the advantages of object-oriented programming, this shared information would only have to be updated in one place.

By declaring in the DTD `<!ENTITY metadata SYSTEM "legalmeta.xml">`, the external file named "legalmeta.xml" is made available to any XML document that is using this DTD. Any text that appears identically in several texts can exist in one file that can be pulled into the document when displayed. This is a huge advantage for the maintenance of Project Gutenberg. The metadata is currently inconsistent across the collection. The content of this metadata, especially the legal notices concerning copyright issues, should be identical on every document. This can easily be achieved with this method of including external files as entities.

Five samples studied and marked up for this project presented a variety of metadata; some that was very general information about Project Gutenberg, and other pieces that were very specific to the document. In general, the metadata is easily divided into logical chunks that are pulled out into separate files. The advantage of doing this is twofold. First, the information in these files would be very easily managed, and second, documents would only have to contain the chunks of metadata that were relevant. The chunks of metadata added to the documents are as follows:

1. generalmeta – This information is very basic, generic information that is present in almost every file. It consisted of some form of "Welcome to the world of plain vanilla ASCII texts…" and some additional information about the copyright information to follow.

2. releasemeta – Found in 4 out of the 5 samples, this file contains information about how the Gutenberg files are released, at what times, etc.
3. gutinfometa – The specific Gutenberg information covering FTP information and ways to volunteer and/or donate to the project.
4. legalmeta – Contains the "small print" from legal counsel about copyright issues and other legal matter related to these files; should be in every text in the collection.
5. experimentmeta – Found only in one of the five samples, contains information about the experiment to put multiple texts in one file.
6. worldlibmeta – This information gives credit to the World Library for providing the text.

Each of the documents has different combinations of these chunks of metadata. They allowed me as the marker to include the simple entity references to the files, without having to markup up this content further. The entity files are all XML files that can contain mixed content, as declared by the <gutmeta> in the DTD, including <title> and <para>.

Results

Discussion - Applying the DTD on Gutenberg Texts

To test the DTD, I marked up a few texts that are representative of some of the different structures and formats. The Dubliners by James Joyce models a fairly typical book structure. A collection of poems by Emily Dickinson contains not only poem structures, but was created as a book containing the collection poems. With the combined DTD, I was able to use a book element with several chapters of poems within

them. Next, Abraham Lincoln's first inaugural address is representative of a government document or speech. Romeo and Juliet by William Shakespeare is a sample of a play format. An Edgar Allan Poe collection of stories is marked up, and contains several types of literary structure. All of these texts are from the Project Gutenberg website. The original document content was not changed, but the metadata entities may vary, only slightly, from the original in an attempt to create boilerplate pieces of metadata.

All marked up documents can be found in Appendix C and online at http://ils.unc.edu/~bluec/gutenbergDTD.

Initial Analysis – Experience Using the Combined DTD

I downloaded Microsoft's XML Validator ("XML Validator," 1999) to use for testing the XML documents, and Edit Pad Lite (Goyvaerts, 2001) for marking up the documents. I chose the texts that I wanted to markup and saved the text files to my hard drive, opened them in Edit Pad, stripped out any metadata that was related only to Project Gutenberg and left any metadata specific to the text, and Saved As an XML file, preserving the original text.

Immediately, I included the opening and closing <guttext> tags around the entire body of the document, including all remaining metadata and the original document. All boilerplate metadata I then enclosed with the <gutmeta> start and end tags and any metadata about the text that remained was enclosed within <markupmeta> tags. This element is detailed below. Then, I immediately decided what top-level element the document required, and put the entire document body within the appropriate tag (i.e., <book>content</book>). Any additional notation after the document body was put into <endgutmeta> tags. At this point, the markup is not well formed or valid, but one can

easily identify the logical structure of the entire Gutenberg text, and the original document body within it.

Two very distinct, yet highly inconsistent chunks of metadata are now clear: gutmeta, and markupmeta. The content of these two categories of metadata are the same in that they contain material that was added to the original document by Project Gutenberg, but differ in that gutmeta contains all of the boilerplate metadata entities that may follow, and markupmeta contains any added info that is specific to this document. While all of this information varies greatly throughout the collection, a few very specific pieces of this metadata are defined as their own elements in the DTD. Certainly, the DTD could have defined elements for every possible piece of information that might appear in the metadata, but only those common pieces, that might possibly be searched on or otherwise processed someday, are defined. Specifically: textnum, gutdate, preparer, gutfilename are available in the DTD as children elements of markupmeta. Every document in the collection has a number, presumably a unique identifier of the document that would be a likely candidate for future processing efforts. The date that the document was added to the collection should be contained within gutdate tags, and if a volunteer is mentioned as preparing, proofreading or scanning the document, this information can be contained within preparer tags. Finally, each document contains information on how the file should be named within the collection, and this information can be marked as gutfilename. In this section of the Gutenberg texts, I did take it upon myself to delete some information that seemed to me redundant with the addition of the XML tags. For example, if the document contained “Etext #,” “Author:,” or “Title:” to

describe the data that followed, after marked up as <textnum> or <title>, the labels seemed unnecessary .

Again, these few tags represent the minimal amount of detail that markupmeta can and should contain, but these pieces of information are helpful for managing the collection, and thus should be marked. Other information such as the title and author of the document may also be included, and the respective tags are available for use within markupmeta as well.

For marking up the main body of the document, using the Search and Replace function of Edit Pad proved invaluable. While each text varied in its structure, quick Replace functions could achieve the majority of the markup of the text. For “The Raven” in the Edgar Allan Poe collection, I searched for end of line markers (/n in some editors, Shift + Enter in Edit Pad Lite) in the Search box, and replaced them with </line>(end of line)<line>. Now the beginning and ending of every line is marked. Then I replaced all instances of <line></line> with <verse>, as I could easily identify the line elements with no content as the place in the original text where a verse started or ended. However, for “The Masque of the Red Death” in the same collection, I searched for all instances with two end of line markers in a row, and replaced them with </para>(end of line)<para>, and then moved the last <para> to the beginning. Each of the texts had obvious patterns that could be discovered and then replaced with markup, but there are no hard and fast rules to follow when doing this. For example, Romeo and Juliet proved very difficult to markup, as stage directions <stagedir> could be found within <speech> or on their own. Some were contained in [brackets], which made them targets for Search and Replace [

with <stagedir>, and] with </stagedir>. Furthermore, to maintain Shakespeare's iambic pentameter each section of speech included <line> tags to maintain the literary structure.

In addition to the content elements and entities, special characters are not parsed properly in XML. For example, an ampersand (&) will not parse correctly as content, as they are typically used to refer to entities. Thus, the marker must enter & in place of the ampersand. The Open eBook Initiative provides an additional DTD entity that provides names for each of the common special characters that can easily be referred to in the document. Perhaps this idea could be incorporated into the Project Gutenberg DTD should the need for it arise.

The best way, in my experience, to check for well formedness, was to simply open the XML document, without a stylesheet, in the Microsoft Internet Explorer 5.0 browser window. While this method does not check for valid XML, it will provide the line number on which the well formedness failed, at which point a marker can return to Edit Pad, and select to view the line numbers, or by selecting Ctrl + G, to get a "Go To Line" box into which you can enter and go straight to the line number causing the error. Document validation can easily be tested with the XML Validator ("XML Validator," 1999), but for the purposes of the Project Gutenberg texts, document well formedness is required, validation is not.

I can foresee that some of these very general and simplistic tasks could be written into scripts that automate this process. The Gutenberg texts might not be the preferred version of the text to run a script on, as the line breaks and paragraph breaks are inconsistent. If another version of the same electronic text can be found that is more consistent, it would be much easier to automate the markup process.

My experience with my prepared DTD only confirmed my earlier suspicions that the DTD needs to be as flexible as possible, with very few unnecessary validity constraints. For example, I had to add <scene> as a child entity to <playbody> when marking up Romeo and Juliet because the prologue of the play is written as a scene, but was not within a <part> or <act> as the original DTD required. I also added <title> as a child element of <scene> as dictated by the Shakespeare play. When marking up the collection of Emily Dickinson plays, I immediately discovered that the constraint of only one title was too restrictive for <poembody>, because poets are renowned for breaking common literary conventions such as this. Many of the Dickinson plays are titled only with a number, such as "IV." whiles others have both a number and a traditional title. Thus, I chose to enclose both pieces of information, if available, within <title> tags, as no other logical structure was available.

It seems that the markup of documents is often subjective. While semantics and structure are often very obvious, the multitude of possible combinations of elements in the Project Gutenberg collection is unpredictable. A marker can include a lot of detail if they perceive it to be necessary, whereas others may not.

Future Steps

To restate the intentions of the project, I feel that this DTD should remain as free and publicly available as the Project Gutenberg staff and volunteers deem necessary. Neither the DTD nor the documents marked up should be copyrighted by anyone other than the original copyright holder, if their rights still apply.

As volunteers markup the Gutenberg documents, the DTD will undoubtedly evolve. The DTD will be dynamic, but not so much as to prevent backward

compatibility. Releases of point versions (i.e., version 1.1) might be released monthly, while new versions (2.0) with fundamental structural changes should only be released annually.

Some decisions on the metadata content entities will have to be settled by the Gutenberg staff, particularly the legal information regarding copyright. However, this should be a very easy step to accomplish even before calling upon volunteers to begin marking up documents. If the entities that I have summarized and extracted from my sample documents do not cover an important chunk of Gutenberg metadata, a manager should determine how to pull it out and define it, so that it can thereafter be updated in only one place.

A long-term possibility of marking up Project Gutenberg texts with XML could provide for foreign language texts and higher bit Unicode character sets. For the management of a collection as large a Project Gutenberg, it might be useful to somehow mark the language of each text. The xml:lang attribute can be included in the XML document to specify the language used. "The intent declared with xml:lang is considered to apply to all attributes and content of the element where it is specified, unless overridden with an instance of xml:lang on another element within that content" (World Wide Web Consortium, 2000). While this capability was not included within this project, the attribute might prove useful at a later date.

Another potential use of this management structure would allow documents to be broken up into smaller, more logical parts. For example, collections of poetry are currently contained in one large file, partially because each file had to contain all of the legal and other metadata, most of which would exceed the size of the document! With

the functionality of entities in XML, each poem could stand on its own, yet still be presented as one collection by pulling them all in to one document as entities. On the other extreme, very large texts could be separated into files based on their chapters, or other structure intrinsic to the text. By doing so, users could download and search very specific texts much more quickly and accurately, without increasing the physical size of the files.

Conclusion

While the ASCII format has been a functional and universally available format for the Project Gutenberg texts for 30 years, the Extensible Markup Language has proven to be a stable and viable format that could provide numerous advantages to the collection at a very low cost. This project includes analyses of similar efforts and their viability for use with Gutenberg, and a suggested path to follow for this effort. I prepared a comprehensive, standalone DTD based upon the efforts of the HTML Writer's Guild DTDs, and marked up a few very different texts from the Gutenberg collection.

The potential for this structure will yield only to the dynamics of the DTD. Should the DTD go through several revisions, evolving too quickly for marker to keep up with and marked up texts to comply with, the future effectiveness of XML with Project Gutenberg could wane. The staff and all volunteers involved with the effort of marking up the texts in this collection must remember to keep this DTD and all marked up documents free of copyright and freely available. While the DTD will likely evolve with the collective experience marking up texts, it must not grow too complex or large to learn and manage.

After thirty years, Project Gutenberg is still functional and thriving. Continuing efforts should always lend to the longevity and lifespan of the texts and the project, and not create a new collection of texts that will soon be obsolete. These principles should guide all future endeavors.

References

Boumphrey, F. (2000a, July). European literature and Project Gutenberg. [On-line]. Cultivate Interactive, 1(3). Available: http://www.cultivate-int.org/issue1/gutenberg.

Boumphrey, F. (2000b, March 7). Gutenberg Project<longish>. Personal Communication: xml-dev@xml.org [On-line]. Available: http://lists.xml.org/archives/xml-dev/200003/msg00232.html.

Boumphrey, F. (2000c, February 11). Re: (Pre)Announce: gutenberg at HWG. Personal Communication: xml-dev@xml.org [On-line]. Available: http://lists.xml.org/archives/xml-dev/200002/msg00264.html.

Cover, R. (2000, November 6). Davenport Group: DocBook DTD. OASIS: The XML Cover Pages: General SGML/XML Applications [On-line]. Available: http://www.oasis-open.org/cover/gen-apps.html.

Cover, R. (2001, February 22). Open eBook initiative. OASIS: The XML Cover Pages [On-line]. Available: http://xml.coverpages.org/openEbook.html.

DeRose, S. & Renear, A. (2000, April 29). Open eBook publication structure. Providence, Rhode Island: Brown University, Scholarly Technology Group [On-line]. Available: http://www.stg.brown.edu/projects/indexcard/displaycard.php3?card=10.

Goyvaerts, J. (2001). EditPad Lite (Version 4.2.0) [Computer software]. Flanders, Belgium: JGsoft. Available: http://www.editpadpro.com/editpadlite.html.

HTML Writers Guild. (2001a). Getting started: Gutenberg at HWG. [On-line]. Available: http://gutenberg.hwg.org/guthowto1.html.

HTML Writers Guild. (2001b). Book DTD's I: Gutenberg at HWG. [On-line]. Available: http://gutenberg.hwg.org/gutdtds1.html..

Mahoney, A., Rydberg-Cox, J. A., Smith, D. A. and Wulfman, C. E. (2000). Generalizing the Perseus XML document manager. Boston, Massachusetts: Tufts University, Project Perseus Digital Library. [On-line]. Available: http://www.perseus.tufts.edu/Articles/exploration.html.

Meggison, D. (2000, March 7). Use TEI. Personal communication [On-line]. Available: http://lists.xml.org/archives/xml-dev/200003/msg00248.html.

Miller, R. (2001, March 5). Nupedia and Project Gutenberg directors answer. Slashdot [On-line]. Available: http://slashdot.org/article.pl?sid=01/03/02/1422244&mode=nested.

Open eBook Forum. (2000). Open eBook publication structure: Specification. [On-line]. Available: http://www.openebook.org/specification.htm.

Open eBook Forum. (1999, September 16). Open eBook publication structure 1.0. [On-line]. Available: http://www.openebook.org/OEB1.html.

Project Gutenberg. (1992, August). History and philosophy of Project Gutenberg. [On-line]. Available: http://promo.net/pg/history.html.

Smith, D. A., Mahoney, A., Rydberg-Cox, J. A. (2000, August). Management of XML documents in an integrated digital library. Boston, Massachusetts: Tufts

University, Project Perseus Digital Library. Paper presented at Extreme Markup Languages 2000: The Expanding XML/SGML Universe, Montreal. [On-line]. Available: http://www.perseus.tufts.edu/Articles/hopper.html.

Spooner, J. G. (1999, August 13). New eBook standard: A best seller? ZDNet News [On-line]. Available: http://www.zdnet.com/zdnn/stories/news/0,4586,2314515,00.html.

St. Laurent, S. (1999). XML: A primer (2nd ed.). Foster City, CA: M&T Books.

University of Virginia Library. TEI guidelines for electronic text encoding and interchange: Note. [On-line]. Charlottesville: University of Virginia, Electronic Text Center. Available: http://etext.lib.virginia.edu/bin/tei-tocs?div=DIV1&id=PF.

University of Virginia Library. TEI guidelines for electronic text encoding and interchange: Underlying principles and intended use, design principles of the TEI scheme. [On-line]. Charlottesville: University of Virginia, Electronic Text Center. Available: http://etext.lib.virginia.edu/bin/tei-tocs?div=DIV2&id=ABDPIU.

Usdin, T. & Graham, T. (1998). XML: Not a silver bullet, but a great pipe wrench. StandardView 6(3), p.125-132.

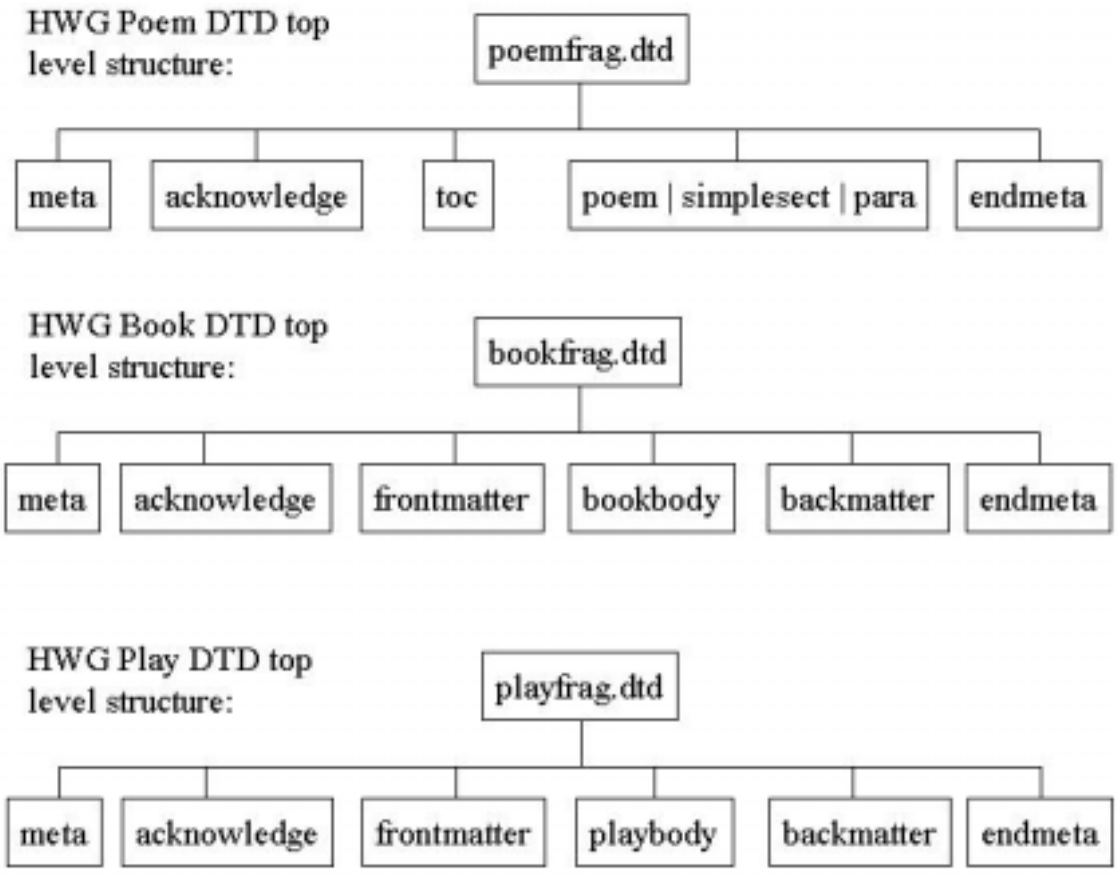
Walsh, N. & Leonard M. (1999). DocBook: The Definitive Guide. Sebastopol, CA: O'Reilly and Associates.

World Wide Web Consortium (W3C). (2000, October 6). Extensible Markup Language (XML) 1.0 (2nd ed.). W3C Recommendation [On-line]. Available: http://www.w3.org/TR/2000/REC-xml-20001006.

XML Validator. [Computer software]. (1999). Redmond, Washington: Microsoft Corporation.

Appendix A

Figure 1. The three main structures from the HTML Writer's Guild DTDs (poem, book and play) contain similar top-level structures, making them likely candidates for combining into one DTD to share these common elements.



Appendix B

The complete DTD created for this project; the DTD should be named gutdtd.dtd and referred to accordingly in the prologue of the XML documents that are adopting this schema. The DTD has been modified from its original formatting for aesthetic reasons, but is available in its originally formatted condition online at http://ils.unc.edu/~bluec/gutenbergDTD/.

```
<!--- START DTD -->

<!ELEMENT guttext
(gutmeta,markupmeta,(book|play|poem|document)*,endgutmeta?)>

<!-- Put all info related to Project Gutenberg in here -->
<!ELEMENT gutmeta (#PCDATA|para|simplesect|title)*>

<!-- Put all text-specific info in here -->
<!ELEMENT markupmeta
(#PCDATA|textnum|preparer|gutdate|gutfilename|title|author|para|simplese
ct)*>

<!ELEMENT endgutmeta (#PCDATA|para|simplesect|title)*>


<!-- ******************** ENTITY DECLARATIONS ********************** -->

<!-- General PG intro, Welcome to the World of...-->
<!ENTITY generalmeta SYSTEM "generalmeta.xml">

<!-- Legal info, should be the same for each -->
<!ENTITY legalmeta SYSTEM "legalmeta.xml">

<!-- General PG info, how texts are released -->
<!ENTITY releasemeta SYSTEM "releasemeta.xml">

<!-- Only found in a few texts, about mulitple items in single file -->
<!ENTITY experimentmeta SYSTEM "experimentmeta.xml">

<!-- About Project Gutenberg, donations, FTP info -->
<!ENTITY gutinfometa SYSTEM "gutinfometa.xml">

<!-- Information and Credit for the World Library -->
<!ENTITY worldlibmeta SYSTEM "worldlibmeta.xml">
```

```
<!ENTITY % dtdattribs
"ref   IDREF  #IMPLIED
 id    ID     #IMPLIED
 type  CDATA  #IMPLIED
 role  CDATA  #IMPLIED
 class CDATA  #IMPLIED"
>

<!ENTITY % inline.class
"|quote|emph|ital|reference|date|place|name|graphic|txterr|mkuperr|misc"
>

<!ENTITY % block.class
"|letter|blockquote|footnote|note|list|deflist|table|blockgraphic">


<!-- ******************** METADATA ELEMENTS ********************** -->

<!-- Put the Gutenberg E-text number in here -->
<!ELEMENT textnum (#PCDATA %inline.class;)*>
<!ATTLIST textnum
  %dtdattribs;
>

<!-- Use for "prepared, proofread, or scanned by -->
<!ELEMENT preparer (#PCDATA %inline.class;)*>
<!ATTLIST preparer
  %dtdattribs;
>

<!-- Use Gutenberg date (when added to collection -->
<!ELEMENT gutdate (#PCDATA %inline.class;)*>
<!ATTLIST gutdate
  %dtdattribs;
>

<!-- Use "This file should be named..." -->
<!ELEMENT gutfilename (#PCDATA %inline.class;)*>
<!ATTLIST gutfilename
  %dtdattribs;
>

<!-- ******************** TOP LEVEL ELEMENTS ********************** -->

<!-- The text can be any of the following elements -->

<!ELEMENT book
(acknowledge?,meta*,frontmatter?,bookbody,backmatter?,endmeta*)>

<!ELEMENT play
(acknowledge?,meta*,frontmatter?,playbody,backmatter?,endmeta*)>

<!ELEMENT poem
(acknowledge?,meta*,toc*,(poembody|simplesect|para)*,endmeta*)>

<!ELEMENT document
(acknowledge?,meta*,frontmatter?,documentbody,backmatter?,endmeta*)>
```

```
<!-- ************** TOP LEVEL ELEMENT DECLARATIONS ***************** -->

<!ELEMENT acknowledge (#PCDATA %inline.class;)*>
<!ATTLIST acknowledge
  %dtdattribs;
>

<!ELEMENT meta EMPTY>
<!ATTLIST meta
   content  CDATA  #REQUIRED
   id       ID     #IMPLIED
>

<!ELEMENT endmeta EMPTY >
<!ATTLIST endmeta
   content  CDATA  #REQUIRED
   id       ID     #IMPLIED
>

<!ELEMENT bookbody (part*|chapter*)>

<!ELEMENT poembody
((prenote|title)*,(author|prose|subtitle|verse|line|para|tune|note|footn
ote)*)>

<!ELEMENT playbody (part|act|scene)*>

<!ELEMENT documentbody ((prenote|title)*,(speech|para|note|misc*))>

<!ELEMENT backmatter ((appendix|index|glossary|biblio|note)*,colophon?)>


<!-- *********** END TOP LEVEL ELEMENT DECLARATIONS ***************** -->


<!-- *********** BEGIN FRONTMATTER ELEMENT DECLARATIONS ************ -->

<!ELEMENT frontmatter (htitlepage|copypage|epigraph|titlepage|

toc|acksect|dedication|preface|prologue|personae|introduction|miscfm)*>

<!ELEMENT htitlepage (#PCDATA|title|subtitle|author|para|poembody|song
%inline.class;)*>
<!ATTLIST htitlepage
  %dtdattribs;
>

<!ELEMENT copypage (#PCDATA|para|poembody|song|note %inline.class;)*>
<!ATTLIST copypage
  %dtdattribs;
>

<!ELEMENT epigraph (#PCDATA|para|poembody|song|note|blockquote
%inline.class;)*>
<!ATTLIST epigraph
  %dtdattribs;
>
```

```
<!ELEMENT titlepage
(#PCDATA|partnum|title|subtitle|author|pubinfo|para|poembody|song|note|l
ine %inline.class;)*>
<!ATTLIST titlepage
  %dtdattribs;
>

<!ELEMENT partnum (#PCDATA %inline.class;)*>
<!ATTLIST partnum
  %dtdattribs;
>

<!ELEMENT pubinfo (#PCDATA|para|line %inline.class;)*>
<!ATTLIST pubinfo
  %dtdattribs;
>

<!ELEMENT toc (#PCDATA|title|subtitle|subsubtitle|item|list|deflist
%inline.class;)*>
<!ATTLIST toc
  toctype (contents|maps|graphics|tables|other) "contents"
  %dtdattribs;
>

<!ELEMENT acksect (#PCDATA|para|poembody|song|note %inline.class;)*>
<!ATTLIST acksect
  %dtdattribs;
>

<!ELEMENT dedication (#PCDATA|title|para|poembody|song|note
%inline.class;)*>
<!ATTLIST dedication
  %dtdattribs;
>

<!ELEMENT preface
((title|chapheader)?,(para|poembody|song|sect1|simplesect
%block.class;)*,endchap?,preauthor?)>
<!ATTLIST preface  %dtdattribs;
>

<!ELEMENT prologue ((title|chapheader)?,(para|poembody|song|simplesect
%block.class;)*,endchap?,preauthor?)>
<!ATTLIST prologue %dtdattribs;
>

<!ELEMENT personae (#PCDATA|title|pgroup|persona|para|note
%inline.class;)*>
<!ATTLIST personae %dtdattribs;
>

<!ELEMENT introduction
((title|chapheader)?,(para|poembody|song|sect1|simplesect
%block.class;)*,endchap?)>
<!ATTLIST introduction  %dtdattribs;
>
```

```
<!ELEMENT miscfm (#PCDATA|para|poembody|song|verse|note
%inline.class;)*>
<!ATTLIST miscfm
  %dtdattribs;
>

<!ELEMENT preauthor (#PCDATA|author %inline.class;)*>
<!ATTLIST preauthor
  %dtdattribs;
>

<!-- ************* END FRONTMATTER ELEMENT DECLARATIONS ************ -->


<!-- ************* BEGIN BODY ELEMENT DECLARATIONS ***************** -->
<!-- BOOKBODY (part|chapter) -->
<!-- PLAYBODY (scene|part|act) -->
<!-- POEMBODY (prenote, title,
author|prose|subtitle|tune|note|footnote|verse) -->
<!-- DOCUMENTBODY (prenote|title, speech|para|note|misc) -->

<!ELEMENT part
(acknowledge?,(titlepage|toc|htitlepage|act|prose)*,chapter*)>
<!ATTLIST part
  %dtdattribs;
>

<!ELEMENT chapter
((title|chapheader)?,(para|poembody|playbody|song|sect1|simplesect|page
%block.class;)*,endchap?)>
<!ATTLIST chapter  %dtdattribs;
>

<!ELEMENT act
(title|scene|speech|poembody|playbody|song|scndesc|stagedir|prose|note)*
>
<!ATTLIST act %dtdattribs;
>

<!ELEMENT prenote (#PCDATA)>
<!ATTLIST prenote %dtdattribs;
>

<!ELEMENT title (#PCDATA %inline.class;)*>
<!ATTLIST title
  %dtdattribs;
>

<!ELEMENT author (#PCDATA %inline.class;)*>
<!ATTLIST author
  %dtdattribs;
>

<!ELEMENT prose (#PCDATA|title|simplesect|para %inline.class;)*>
<!ATTLIST prose %dtdattribs;
>
```

```
<!ELEMENT subtitle (#PCDATA %inline.class;)*>
<!ATTLIST subtitle
  %dtdattribs;
>

<!ELEMENT tune (#PCDATA)>
<!ATTLIST tune %dtdattribs;
>

<!ELEMENT verse (title|subtitle|line|note)*>
<!ATTLIST verse
  %dtdattribs;
>


<!-- ************ BEGIN LOWERLEVEL ELEMENT DECLARATIONS ************ -->

<!ELEMENT subsubtitle (#PCDATA %inline.class;)*>
<!ATTLIST subsubtitle
  %dtdattribs;
>

<!ELEMENT chapheader
(title|subtitle|chapnum|chapsummary|blockquote|para|note)*>
<!ATTLIST chapheader
  %dtdattribs;
>

<!ELEMENT chapnum (#PCDATA %inline.class;)*>
<!ATTLIST chapnum
  %dtdattribs;
>

<!ELEMENT chapsummary (#PCDATA %inline.class;)*>
<!ATTLIST chapsummary
  %dtdattribs;
>

<!ELEMENT endchap (para %block.class;)*>
<!ATTLIST endchap  %dtdattribs;
>

<!ELEMENT attrib (#PCDATA %inline.class;)*>
<!ATTLIST attrib
  %dtdattribs;
>

<!ELEMENT caption (#PCDATA %inline.class;)*>
<!ATTLIST caption
  %dtdattribs;
>

<!ELEMENT song (#PCDATA|title|subtitle|verse|line|note|footnote
%inline.class;)*>
<!ATTLIST song
  %dtdattribs;
>
```

```
<!ELEMENT line (#PCDATA %inline.class;)*>
<!ATTLIST line
  %dtdattribs;
>

<!ELEMENT para (#PCDATA|title %inline.class;)*>
<!ATTLIST para
  %dtdattribs;
>

<!ELEMENT simplesect (title,(subtitle|para|poembody|song
%block.class;)*)>
<!ATTLIST simplesect
  %dtdattribs;
>

<!ELEMENT sect1 (title,(sect2|simplesect|para|poembody|song
%block.class;)*)>
<!ATTLIST sect1
  %dtdattribs;
>

<!ELEMENT sect2 (title,(sect3|simplesect|subtitle|para|poembody|song
%block.class;)*)>
<!ATTLIST sect2
  %dtdattribs;
>

<!ELEMENT sect3 (title,(sect4|simplesect|subtitle|para|poembody|song
%block.class;)*)>
<!ATTLIST sect3
  %dtdattribs;
>

<!ELEMENT sect4 (title,(simplesect|subtitle|para|poembody|song
%block.class;)*)>
<!ATTLIST sect4
  %dtdattribs;
>

<!ELEMENT page (#PCDATA %inline.class;)*>
<!ATTLIST page
  %dtdattribs;
>


<!-- start play elements -->

<!ELEMENT speech (#PCDATA|speaker|stagedir|song|poembody|line
%inline.class;)*>
<!ATTLIST speech %dtdattribs;
>

<!ELEMENT scene
(title|speech|poembody|song|scndesc|stagedir|prose|note)*>
<!ATTLIST scene %dtdattribs;
>
```

```
<!ELEMENT speaker (#PCDATA %inline.class;)*>
<!ATTLIST speaker %dtdattribs;
>

<!ELEMENT scndesc (#PCDATA %inline.class;)*>
<!ATTLIST scndesc %dtdattribs;
>

<!ELEMENT stagedir (#PCDATA  %inline.class;)*>
<!ATTLIST stagedir %dtdattribs;
>

<!ELEMENT pgroup (#PCDATA|title|persona|para|note %inline.class;)*>
<!ATTLIST pgroup  %dtdattribs;
>

<!ELEMENT persona (#PCDATA|actor|actress %inline.class;)*>
<!ATTLIST persona  %dtdattribs;
>

<!ELEMENT actor (#PCDATA %inline.class;)*>
<!ATTLIST actor %dtdattribs;
>

<!ELEMENT actress (#PCDATA %inline.class;)*>
<!ATTLIST actress %dtdattribs;
>

<!-- end play elements -->


<!-- ************* BEGIN BACKMATTER ELEMENT DECLARATIONS *********** -->

<!ELEMENT index (title|item|list|deflist|note)*>
<!ATTLIST index
  indtype (contents|authors|firstlines|tables|other) "contents"
  %dtdattribs;
>

<!ELEMENT glossary (title|item|list|deflist|note)*>
<!ATTLIST glossary
  %dtdattribs;
>

<!ELEMENT biblio (title|item|list|deflist|note)*>
<!ATTLIST biblio
  %dtdattribs;
>

<!ELEMENT appendix
((title|chapheader)?,(para|poembody|song|sect1|simplesect
%block.class;)*,endchap?)>
<!ATTLIST appendix  %dtdattribs;
>
```

```
<!ELEMENT colophon (#PCDATA|para|poembody|song %inline.class;)*>
<!ATTLIST colophon
  %dtdattribs;
>


<!-- the block elements -->

<!-- letter elements, a block element -->

<!ELEMENT letter
(address|to|from|salut|sig|title|subtitle|para|poembody|song|line|note)*
>
<!ATTLIST letter
  %dtdattribs;
>

<!ELEMENT address (#PCDATA|para|line %inline.class;)*>
<!ATTLIST address
  %dtdattribs;
>

<!ELEMENT to (#PCDATA|para|line %inline.class;)*>
<!ATTLIST to
  %dtdattribs;
>

<!ELEMENT from (#PCDATA|para|line %inline.class;)*>
<!ATTLIST from
  %dtdattribs;
>

<!ELEMENT salut (#PCDATA|para|line %inline.class;)*>
<!ATTLIST salut
  %dtdattribs;
>

<!ELEMENT sig (#PCDATA|para|line %inline.class;)*>
<!ATTLIST sig
  %dtdattribs;
>
<!-- end letter elements-->

<!ELEMENT blockquote (title?,(para|poembody|song)*,attrib?)>
<!ATTLIST blockquote
  %dtdattribs;
>

<!ELEMENT footnote (#PCDATA %inline.class;)*>
<!ATTLIST footnote
  %dtdattribs;
>
```

```
<!ELEMENT note (#PCDATA %inline.class;)*>
<!ATTLIST note
  %dtdattribs;
>

<!-- list elements, a block element -->

<!ELEMENT list (title?,(list|item)*)>
<!ATTLIST list
  %dtdattribs;
>

<!ELEMENT item (#PCDATA|para|poembody|song|simplesect %block.class;
%inline.class;)*>
<!ATTLIST item
  %dtdattribs;
>

<!ELEMENT deflist (title?,(item,desc?,def*)*)>
<!ATTLIST deflist
  %dtdattribs;
>

<!ELEMENT desc (#PCDATA %inline.class;)*>
<!ATTLIST desc
  %dtdattribs;
>

<!ELEMENT def (#PCDATA %inline.class;)*>
<!ATTLIST def
  %dtdattribs;
>

<!-- end list elements -->


<!-- table elements, a block element -->

<!ELEMENT table (title?,row*,caption?)>
<!ATTLIST table
  %dtdattribs;
>

<!ELEMENT row (cell)*>
<!ATTLIST row
  %dtdattribs;
>

<!ELEMENT cell (#PCDATA %block.class; %inline.class;)*>
<!ATTLIST cell
  %dtdattribs;
>

<!--end table elements-->
```

```
<!--graphics-->
<!-- an inline element, description required -->
<!ELEMENT graphic EMPTY >
<!ATTLIST graphic
  desc CDATA #REQUIRED
  href CDATA #REQUIRED
  %dtdattribs;
>

<!-- a block element -->
<!ELEMENT blockgraphic (title?,graphic,caption?)>
<!ATTLIST blockgraphic
  %dtdattribs;
>


<!-- the inline elements-->

<!ELEMENT quote (#PCDATA %inline.class;)*>
<!ATTLIST quote
  %dtdattribs;
>

<!ELEMENT emph (#PCDATA %inline.class;)*>
<!ATTLIST emph
  %dtdattribs;
>

<!ELEMENT ital (#PCDATA %inline.class;)*>
<!ATTLIST ital
  %dtdattribs;
>

<!ELEMENT reference (#PCDATA %inline.class;)*>
<!ATTLIST reference
  %dtdattribs;
>

<!ELEMENT date (#PCDATA %inline.class;)*>
<!ATTLIST date
  %dtdattribs;
>

<!ELEMENT place (#PCDATA %inline.class;)*>
<!ATTLIST place
  %dtdattribs;
>

<!ELEMENT name (#PCDATA %inline.class;)*>
<!ATTLIST name
  %dtdattribs;
>

<!-- inline error elements, use to enclose or note potential errors in
text -->
```

```
<!ELEMENT txterr (#PCDATA)*>
<!ATTLIST txterr
  explain  CDATA  #IMPLIED
  %dtdattribs;
>

<!--an explanation is required-->
<!ELEMENT mkuperr EMPTY>
<!ATTLIST mkuperr
  explain  CDATA  #REQUIRED
  %dtdattribs;
>

<!ELEMENT misc (#PCDATA %inline.class;)*>
<!ATTLIST misc
  %dtdattribs;
>
<!-- end inline elements -->

<!-- END DTD -->
```

Appendix C

Each of the documents from the Project Gutenberg collection used for testing with the gutdtd DTD are online, in their entirety at http://ils.unc.edu/~bluec/gutenbergDTD/; however, due to their length, each document is only presented as a screen shot.

Figure C1. "The Raven," by Edgar Allan Poe marked up using the gutdtd DTD without a stylesheet; use of poem, poembody, title, several verse and line elements are visible in this Figure. The complete document is available online at http://ils.unc.edu/~bluec/gutenbergDTD/docs/1epoe10.xml.

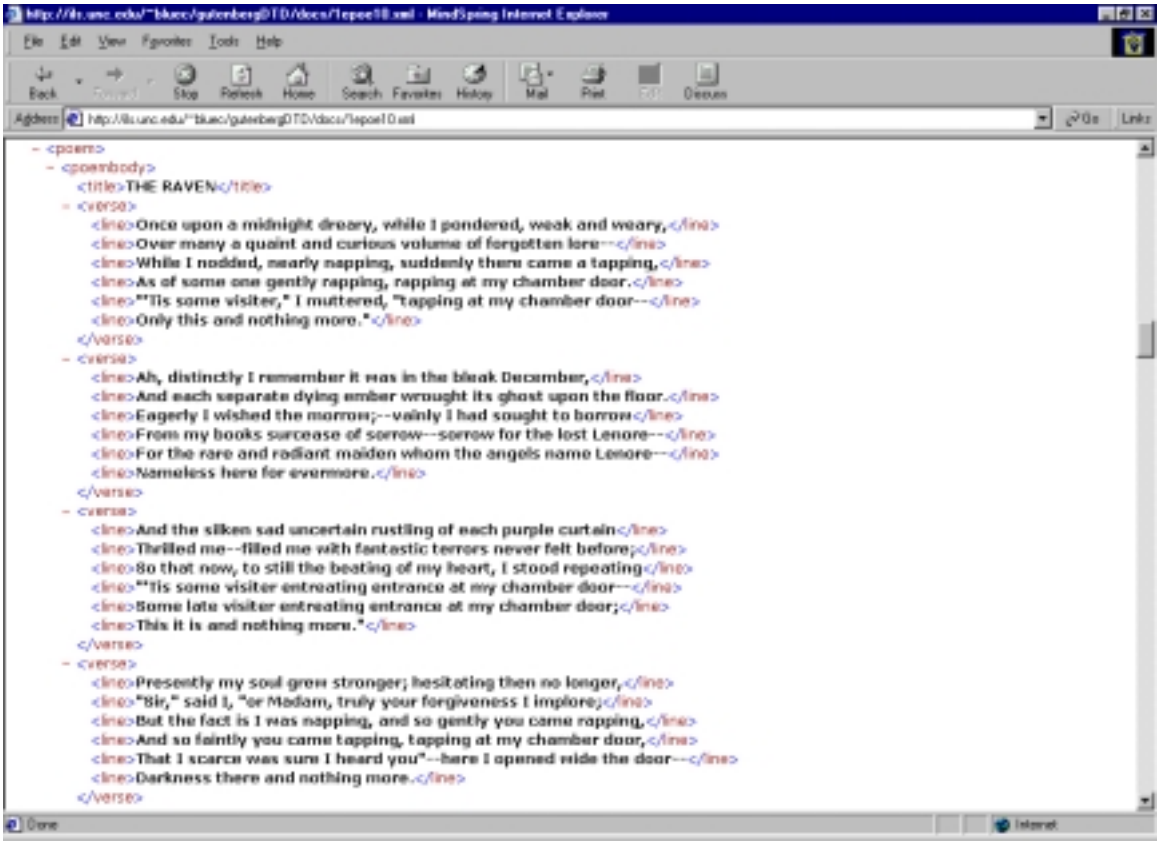


Figure C2. A collection of Emily Dickinson poems marked up using the gutdtd DTD without a stylesheet. This section of the text shows the <markupmeta> element, and its use of the title, author, gutdate, textnum, gutfilename and preparer tags to mark the metadata added to the text by the Gutenberg staff and volunteers, as well as the book frontmatter and titlepage information and preface. The complete document is available online at http://ils.unc.edu/~bluec/gutenbergDTD/docs/1mlyd10.xml.

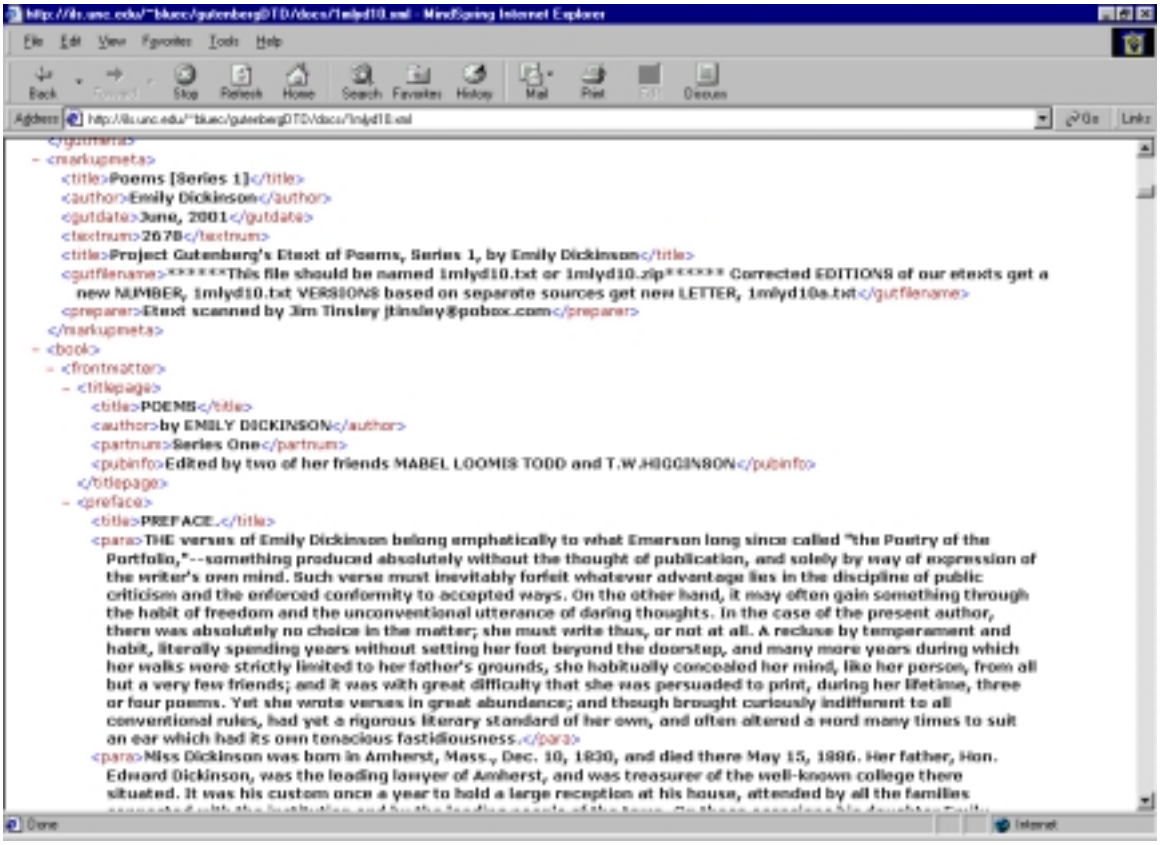


Figure C3. William Shakespeare's Romeo and Juliet was a test document for marking up a play with the DTD. This figure demonstrates the complexity of some of the texts in the Gutenberg collection, here stage directions (<stagedir>) can appear anywhere throughout the text, and the iambic pentameter of the famous Shakespeare play must be maintained with <line> tags. On-line at: http://ils.unc.edu/~bluec/gutenbergDTD/docs/1ws1610.xml.

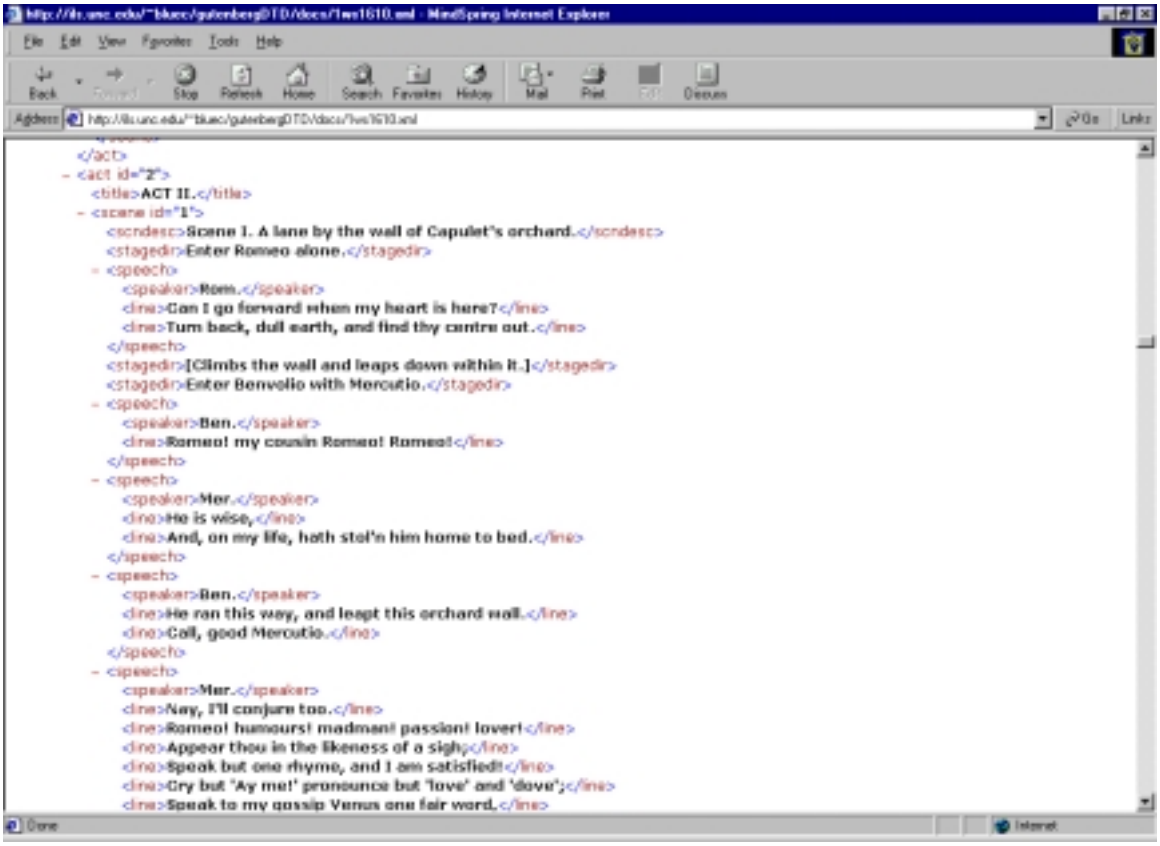


Figure C4. The Dubliners by James Joyce is typical of a book structure. In the section provided below, the opening book tag prefaces the book's frontmatter with a table of contents,<toc toctype = "contents">, and the start of the bookbody and chapter. The complete XML document is available online at http://ils.unc.edu/~bluec/gutenbergDTD/docs/dblnr10.xml.

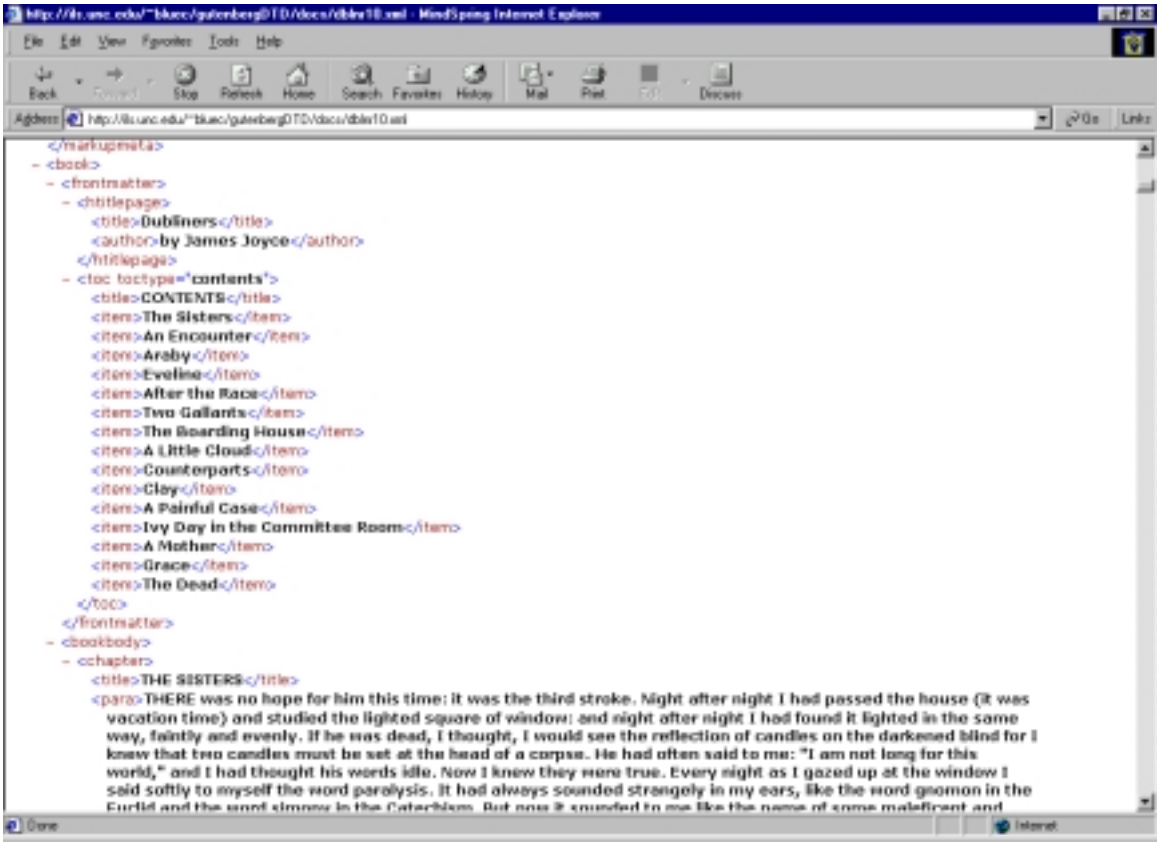


Figure C5. This screen shot is the source of Abraham Lincoln's first inaugural address, which shows the XML document without the browser's interpretation. Noteworthy portions of this Figure are the prologue, the entity declarations within the <gutmeta> tags. Each chunk of metadata that appeared to be boilerplate information was deleted and replaced with the appropriate entity reference.

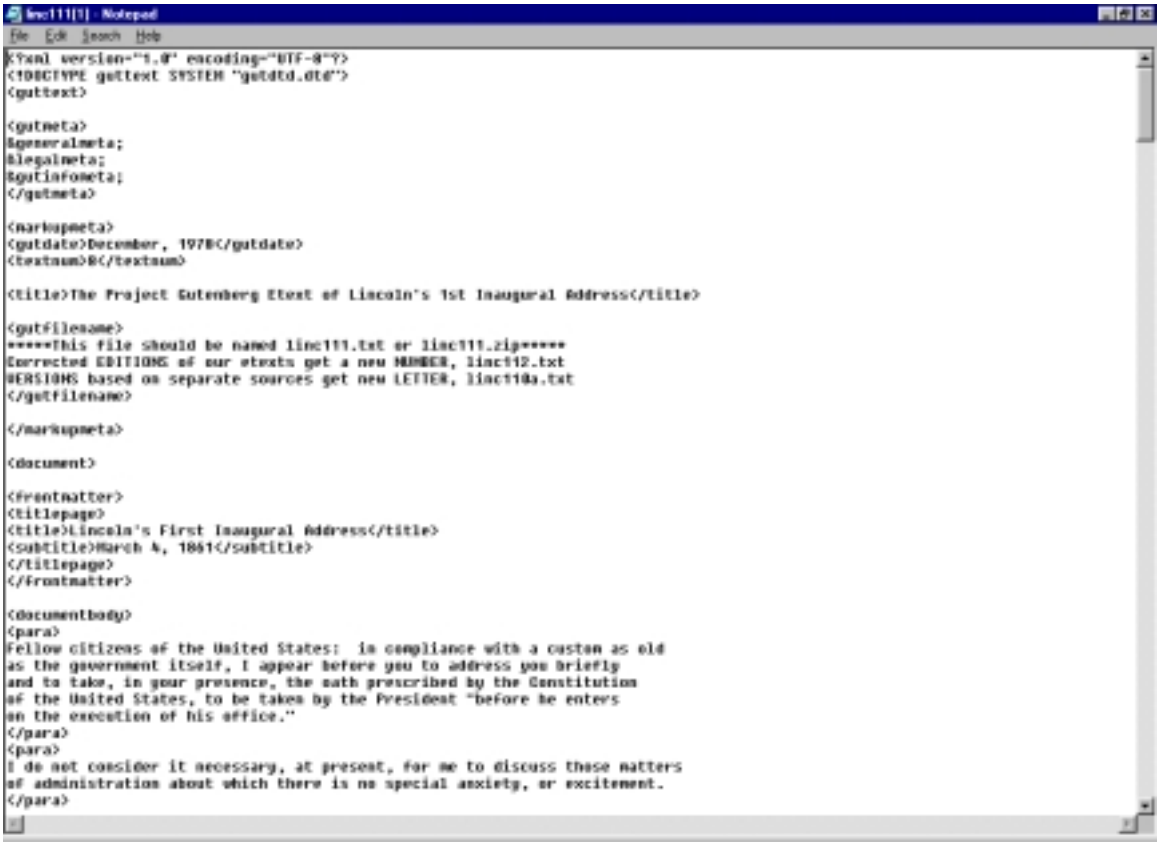
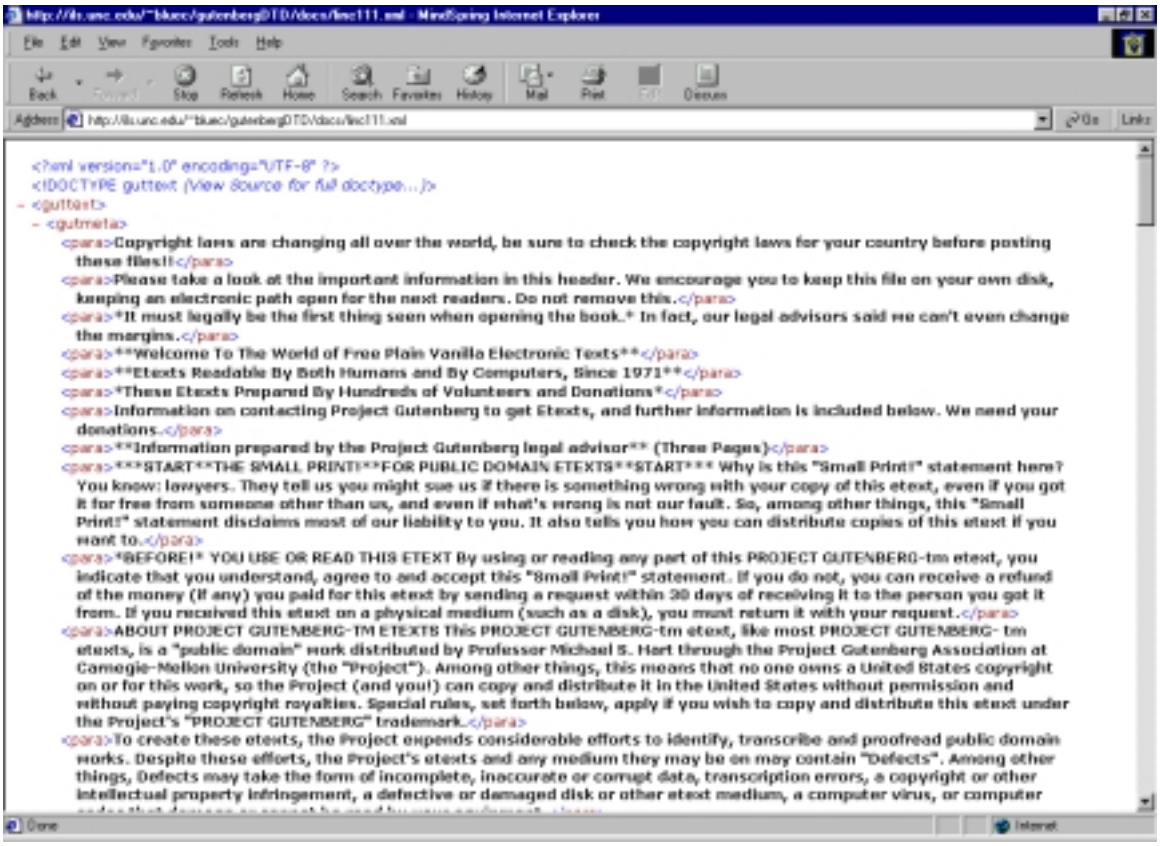


Figure C6. The browser rendered version of Abraham Lincoln's first inaugural address. As demonstrated in this Figure, the syntax of the entity references is not visible, instead they pull in their respective boilerplate metadata content. Available online at http://ils.unc.edu/~bluec/gutenbergDTD/docs/linc111.xml.



Appendix D

The information in this section is a Word revision of the online tutorial for this project that details the components of XML and DTDs in general, and specifics of the DTD prepared for this project, including an index of the elements available in the DTD with brief descriptions. Available online at: http://ils.unc.edu/~bluec/gutenbergDTD/tutorial.html.

Marking up Project Gutenberg Texts with the guttext DTD and XML

The following sections cover the basics of XML, DTDs, and some suggestions on how to proceed with marking up the Project Gutenberg texts with the gutdtd DTD.

Components of XML

The components of XML will look familiar to HTML users: tags, elements and attributes. A tag is a piece of markup, an opening tag <title>, and a closing tag </title>. Tags are used in the composition of elements. This method of markup is used to create XML documents. XML documents can be well formed and valid. A well formed document is syntactically correct and can be interpreted by the computer but does not refer to a Document Type Definition (DTD). Syntactical correctness includes:

- Utilizing a root element;
- Providing closing tags for all opening tags;
- Placing quotes around all attribute values;
- Ensuring the same case is maintained throughout the tags.

A valid XML document is well formed but also complies with the requirements of a DTD. A DTD can be part of the XML document, or an external DTD referred to by the XML document.

The collaboration of the document marked up with XML and the DTD provides content for the browser to interpret and display. The XML document must start with a declaration such as <?xmlversion 1.0?> to tell the browser the version of XML the document is using. Next, if an external DTD is being used, the <!DOCTYPE topelement SYSTEM "file_name.dtd"> announces which DTD is to be used, making up the prolog of the XML document, or "the glue that binds DTDs to the code that applies to them" (St. Laurent, 1999, p.117), and contains the physical location of the DTD as well as whether is it a system or a public DTD. A system DTD is one that has been developed for a particular Web site or organization, while a public DTD has been developed for use by many organizations, mainly for interoperability.

The elements and attributes comprise the logical structure of the XML document. The DTD defines the available elements and attributes, and these specifications can be incorporated by a single XML document or document groups. The contents of the XML documents are not formatted; formatting requires the use of a stylesheet such as Cascading Style Sheets (CSS) or Extensible Style Language (XSL). In the XML document, a line is added to the prologue that contains a reference to the stylesheet such as, <?xml-stylesheet href="xml.css" type="text/css"?>. Each element of the DTD and hence the resulting XML document are displayed according to formatting qualities such as display, font-size, font-weight, and color. The display style determines whether the

contents of an element will be displayed as a separate paragraph or within an existing paragraph. Font-size, weight and color all refer to the style of the text.

Components of a DTD

Elements

The element, declared by <!ELEMENT name data>, defines a storage unit in which data will be held. The "data" portion defines the type of data an element can contain, including other elements and attributes (World Wide Web Consortium, 2000). The content of an element can be of four types:

1. Mixed content - usually declared with parsed character data (#PCDATA). #PCDATA allows any text or any child elements to appear without placing any restrictions on them.
2. List of elements - with rules setting which are required and the order they appear in, how many times they are allowed to be used
3. EMPTY - no content, may have attributes but that this all
4. ANY - is acceptable, but not recommended.

There are several other notations that appear throughout this and other DTDs that identify what content or type of content an element may hold, as well as what order and frequency the elements can appear in. The following list summarizes the symbols used:

- ELEMENT - alone signifies that the element can appear once and only once.
- ELEMENT+ - is required to appear at least once, but can appear many times.
- ELEMENT* - signifies that any number can appear, including zero
- ELEMENT? - this element is optional, but can only appear once.
- ELEMENT, ELEMENT - these elements must appear in this order

- ELEMENT | ELEMENT - or
- (ELEMENT | ELEMENT)* - parentheses group elements; this group suggests that many of either element could appear in any sequence.

Attributes

<!ATTLIST name values> declares additional attributes for an element, and for clarity should appear immediately beneath the element that they describe, but do not have to. They mainly function just for processing, adding additional information that could be used. Attributes can be required (#REQUIRED), optional (#IMPLIED), have a fixed value (#FIXED value), or have a default value. Attributes can have the following types:

- CDATA - character data
- ID - unique value
- IDREF - refers to an ID value somewhere else within the document
- ENTITY(s) - correspond to the name of an external entity
- NMTOKEN(s) - like CDATA, but restricted to letters, digits, periods, dashes, underscores, or colons.
- (value | value) - the value of the attributes must be on of the ones listed
- NOTATION(value | value) - value of the attribute must match the name of one of the NOTATION names listed. CDATA, ID and enumerated types are the most common (St. Laurent, 1999, p.138).

Entities

Block and inline entities are used for parts of text that can appear anywhere, and are mainly defined as inline or block based upon their general appearance within the text. Inline entities are ones that appear within a line of text, but can appear anywhere within

the text and therefore do not have specific requirements for use within the elements. Block entities are used in a similar manner, but appear within a separate paragraph or aesthetic chuck of a text. An example of an inline element is a 'date', and a block entity might be a 'blockquote' or a 'table.' Any references similar to:

```
<!ELEMENT acknowledge (#PCDATA | %inline.class;)*>
```

declares that the acknowledge element can contain PCDATA any of the elements, in any order, any number of times, that were declared within the inline.class entity.

Entities are also used to pull out all of the Gutenberg-related metadata that is the same across all of the texts into a separate entity or entities, that would be stored in separate files and referenced or pulled into the XML document. By declaring in the DTD:

```
<!ENTITY metadata SYSTEM "legalmeta.xml">
```

the external file named "legalmeta.xml" is made available to any XML document that is using this DTD. Any boilerplate text that appears identically in several texts can exist in one file which can be pulled into the document when displayed.

A few other entities are also available for use throughout the DTD. First, the entity "dtdattribs" declares five attributes that can easily be added to any element, and updated in one place for the entire DTD. These attributes are: ref, id, type, role, and class.

Marking-up a Text

All Project Gutenberg texts that will be marked up with this DTD will start with the <guttext> tag. This tag simply represents the beginning and ending of the markup, inside which all of the other elements and markup must reside. This is an easy first step to take for existing and unmarked etexts, as well as for texts that are being newly typed or scanned.

While the fundamental content of any Gutenberg text is the original book, play, poem or document, there is additional text that is added by the Project Gutenberg texts. Therefore, the <guttext> element can contain 4 main elements that must appear in sequence if they appear. These are <gutmeta>, <markupmeta>, either <book, poem, play, or document> and <endgutmeta>. Think of this topmost level as the entire Gutenberg text, not just the book or poem it might contain. Setting up these first tags sets the container for all of the content of the text. The main document content will always appear within the opening and closing.

Start by adding opening and closing <guttext> tags around the entire body of the document, including all remaining metadata and the original document. All boilerplate metadata should be enclosed with the <gutmeta> start and end tags, and any metadata about the text within <markupmeta> tags. Then, decide what top-level element the document requires, and put the entire document body within the appropriate tag (i.e., <book>content</book>). Put any additional notation after the document body into <endgutmeta> tags. At this point, the markup is not well formed or valid, but one can easily identify the logical structure of the entire Gutenberg text, and the original document body within it.

Two very distinct, yet highly inconsistent chunks of metadata are now clear: gutmeta, and markupmeta. The content of these two categories of metadata are the same in that they contain material that was added to the original document by Project Gutenberg staff, but differ in that gutmeta contains all of the boilerplate metadata entities that may follow, and markupmeta contains any added info that is specific to this document. While all of this information varies greatly throughout the collection, a few

very specific pieces of this metadata are defined as their own elements in the DTD, specifically: textnum, gutdate, preparer, gutfilename. Every document in the collection has a number, presumably a unique identifier of the document that would be a likely candidate for future processing efforts. The date that the document was added to the collection should be contained within gutdate tags, and if a volunteer is mentioned as preparing, proofreading or scanning the document, this information can be contained within preparer tags. Finally, each document contains information on how the file should be named within the collection, and this information can be marked as gutfilename. In this section of the Gutenberg texts, it is suggested that some information is deleted; if the document contains “Etext #,” “Author:,” or “Title:” to describe the data, after marked up as <textnum> or <title>, the labels are unnecessary .

So far, this is fairly simple. To review, we have:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE guttext SYSTEM "my_dtd.dtd">
<guttext>
<gutmeta>entity reference</gutmeta>
<markupmeta>test</markupmeta>

<book, poem, play or document>
The remainder of this tutorial will focus on this main content section
and how to mark it up.

</book, poem, play or document>
</guttext>
```

The next step for a marker is the hardest one, and requires one to think about the substance of the text, and the roles and functions of the content, in order to mark it up properly. In the <guttext> declaration, the DTD allows for the marker to add a book, play, poem or document. These elements are optional, and can appear in any order, as many times as necessary. They should each contain a distinct work. In other words, a collection of poems could contain dozens of <poem> elements, but they would all clearly represent the beginning and ending of one poem in the collection. Each of the four has a distinct set of elements that can appear within them.

1. book: acknowledge, meta, frontmatter, bookbody, backmatter, endmeta
2. poem: acknowledge, meta, toc, poembody, simplesect, para, endmeta
3. play: acknowledge, meta, frontmatter, playbody, backmatter, endmeta
4. document: acknowledge, meta, frontmatter, documentbody, backmatter, endmeta

While quite a few of these sets of elements overlap, they will help to define the structure of the texts further. Notice also that each of these three lists is comma-delimited in the DTD, signifying that they must appear in this sequence if they appear. The selection of this top-level element should be easy to determine (i.e., the text is either a book or a poem); the document entity can be used for miscellaneous government documents or speeches.

So how does one decide which elements to use and how to use them? My opinion is to think abstractly at first to find the major sections of the text. For example, skim a large book to see if it has parts and chapters or just chapters. Take notice of the front and back matter of the book; is there a table of contents or an index, a glossary or an

appendix? For the purposes of Project Gutenberg, display and searching needs will probably make use of these tags the most.

With these larger sections in mind, markup can begin, and the smaller details can be taken into consideration. The remainder of this tutorial provides and index and a description of each of the elements available in this DTD.

The Elements

All of the elements in the gutdtd DTD can be found indexed here by the following categories: metadata elements, top-level elements, frontmatter elements, main body elements, lower-level elements, play elements, backmatter elements, block elements, inline elements.

These categories are only intended to help the marker visualize the main types of content and elements of a text, and do not restrict that element to a particular category or use. The nature of the DTD allows elements to refer to each other and share a set of elements. Some of the elements contain only other elements, and thus may be considered more of a "top-level" or "main body" element, and yet others might be considered leaf elements in that they contain PCDATA or generic child elements (such as para or line) so appear as "lower-level" elements.

Since the DTD is a revision of the HTML Writer's Guild DTDs designed for use with the Gutenberg texts, much of the credit for this list must be attributed to them. The HTML Writer's Guild based most of their structure on the "Parts of the Book" from The Chicago Manual of Style. Some elements are unique to this project. The following lists contain information about the intended use of elements that may be unclear, and the types of data and sub-elements that can appear within them. However, to determine the

requirements of the element in regards to the order or number of times that its sub-elements can appear, refer to the complete DTD.

The Metadata Elements

1. gutmeta – “Gutenberg metadata,” use this tag to enclose all boilerplate metadata that is added to the texts in the collection by the Project Gutenberg staff and volunteers. Usually will contain only entity references to the external boilerplate files, but can also contain PCDATA, para, simplesect, and title elements.
2. markupmeta - all additional metadata added by the Project Gutenberg staff that is unique to this particular text, should be enclosed within this tag; can contain PCDATA, textnum, preparer, gutdate, gutfilename, title, author, para, or simplesect elements.
3. textnum - the "Etext #___" found within the metadata of the Gutenberg collection; can contain PCDATA, any of the inline elements, or the DTD attributes.
4. preparer - use for information about who may have prepared, proofread or scanned the document for the collection; can contain PCDATA, any of the inline elements, or the DTD attributes.
5. gutdate – “Gutenberg date,” use for the date found in the metadata, can contain PCDATA, any of the inline elements, or the DTD attributes.
6. gutfilename - use for any information about how the file should be named, can contain PCDATA, any of the inline elements, or the DTD attributes.
7. endgutmeta – “End Gutenberg metadata,” any additional information that is found at the end of the text can be enclosed within these tags, can contain PCDATA, or the para, simplesect, or title elements.

The Top-level Elements

1. guttext – "Gutenberg text," the very, highest, absolutely required element in that should appear as the very first and very last tag in the XML document, enclosing all pieces of the text - both the information added by Project Gutenberg and the original document; can contain gutmeta, markupmeta, book, play, poem and/or document, and endgutmeta elements.
2. book - the main book element, can contain: acknowledge, meta, frontmatter, bookbody, backmatter, endmeta.
3. play - the main play element, can contain: acknowledge, meta, frontmatter, playbody, backmatter, endmeta.
4. poem - the main poem element, can contain: acknowledge, meta, toc, poembody, simplesect, para ,endmeta.
5. document - the main document element, for any other text such as a speech or government document that does not make sense as a book, play or poem; can contain acknowledge, meta, frontmatter, documentbody, backmatter, endmeta.
6. acknowledge - ? for the main book, poem, document and play elements. It can contain PCDATA, any of the inline elements and the DTD attributes. Common usage of this element would likely be any recognition given to others at the beginning of the text.
7. meta - * for each of book, poem, document and play, is EMPTY but has the content and id attributes.
8. endmeta - * for each of book, poem, document and play, EMPTY but has the content and id attributes.

9. frontmatter - ? for book, document and play, not available to poem. Frontmatter contains only sub-elements, not actual content, and should be used to enclose any information that is contained before the actual body of the document starts. These available sub-elements include: htitlepage, copypage, epigraph, titlepage, toc, acksect, dedication, preface, prologue, personae, introduction, or miscfm.
10. backmatter - ? for book, document and play, not available to poem. Back matter contains only sub-elements. This element should be used to enclose any information that is contained after the actual body of the document ends. The sub-elements available within backmatter are: appendix, index, glossary, biblio, note and colophon.
11. bookbody - required for book, can contain as many parts and chapters as needed.
12. poembody - required for poem, can contain prenote, title, author, prose, subtitle, tune, note, footnote and verse.
13. playbody - required for play, can contain as many parts and acts as needed.
14. documentbody - required for document, can contain prenote, title, speech, para, note, misc.

Frontmatter Elements

While these sub-elements are not restricted for use within the frontmatter parent element, the frontmatter parent element has provided the context for these initially.

1. htitlepage - can contain PCDATA, any of the inline elements or DTD attributes, or the following sub-elements: title, subtitle, author, para, poem or song. This element can otherwise be thought of as a "half" titlepage, or a page that only contains the title.

2. copypage - can contain PCDATA, any of the inline elements or DTD attributes, or the following sub-elements: para, poem, song or note. Typically this it the copyright page of a text.
3. epigraph - can contain PCDATA, any of the inline elements or DTD attributes, or the following sub-elements: para, poem, song, note or blockquote. Suggested use of this element if for any quote or statement found before the text begins.
4. titlepage - can contain PCDATA, any of the inline elements or DTD attributes, or the following sub-elements: partnum, title, subtitle, author, pubinfo, para, poem, song, note or line. The main title page of a book usually contains the publication information (pubinfo, see below).
5. pubinfo - can contain PCDATA, any of the inline elements or DTD attributes, or the following sub-elements: para or line. This element is available to the entire DTD, but is listed here in the context of the title page reference.
6. toc - can contain PCDATA, any of the inline elements or DTD attributes, or the following sub-elements: title, subtitle, subsubtitle, item, list or deflist. Also contains an additional attribute "toctype" that allows the marker to specify whether this is a table of contents, maps, graphics, tables or other type, defaults to ‘contents.’ Usually tables of contents are irrelevant in etexts as page numbers are meaningless.
7. acksect - can contain PCDATA, any of the inline elements or DTD attributes, or the following sub-elements: para, poem, song or note.

8. dedication - can contain PCDATA, any of the inline elements or DTD attributes, or the following sub-elements: title, para, poem, song, or note. Most often, dedications are present on their own page.
9. preface - can contain PCDATA, any of the block elements or DTD attributes, or the following sub-elements: a title or chapheader, any of para, poem, song, sect1, simplesect, and endchap or preauthor. Prefaces usually include an author's motivation for writing the book or the sources and assistance they might have received in writing it.
10. prologue - can contain PCDATA, any of the block elements or DTD attributes, or the following sub-elements: a title or chapheader, any of para, poem, song, simplesect, and endchap or preauthor.
11. personae - can contain PCDATA, any of the inline elements or DTD attributes, or the following sub-elements: title, pgroup, persona, para or note.
12. introduction - can contain PCDATA, any of the block elements or DTD attributes, or the following sub-elements: a title or chapheader, any of para, poem, song, sect1, simplesect, and endchap. Introductions are text that is related directly to the main body of the text, but should be read before reading the body of the text.
13. miscfm – “miscellaneous frontmatter,” can contain PCDATA, any of the inline elements or DTD attributes, or the following sub-elements: para, poem, song or note. This category is reserved for any additional frontmatter not covered by another element, like a list of abbreviations that might be used in the text, an editorial or chronology that helps to set up, or give context to the main body.

14. preauthor - can contain PCDATA, any of the inline elements or DTD attributes, or the author sub-element. A preauthor element would be used if someone other than the author of the text wrote the preface, etc. While this element is available to the entire DTD, it is included here in context of the frontmatter elements.

Main Body Elements

The following elements are ones that are usually in context with the bookbody, playbody or poembody main text body content.

1. part - * for bookbody and playbody, can contain PCDATA, any of the DTD attributes, or the following sub-elements: acknowledge, titlepage, toc, htitlepage, act, prose or chapter. Sometimes large books have parts with the chapters inside them, the sub-elements represent, for example, a title page specific to that part.
2. chapter - can contain PCDATA, any of the block elements or DTD attributes, or the following sub-elements: title or chapheader, para, poem, song, sect1, simplesect, page and endchap.
3. act - can contain PCDATA, any of the DTD attributes, or the following sub-elements: title, scene, speech, poem, song, scndesc, stagedir, prose or note. Used to signify the acts in a play.
4. prenote - a leaf element that can contain PCDATA or any of the DTD attributes. Use for an introductory note.
5. partnum - a leaf element that can contain PCDATA, the inline elements or any of the DTD attributes. Use on titlepage for volume or part numbers of the text.
6. title - a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.

7. author - a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.
8. prose - a mixed content element, can contain PCDATA, any of the inline elements of DTD attributes, or the following sub-elements: title, simplesect or para.
9. subtitle - a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.
10. tune - a leaf element that can contain PCDATA or any of the DTD attributes.
11. verse - can contain any of the DTD attributes or the following sub-elements: title, subtitle, line, or note.

Lower-level Elements

1. subsubtitle - a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.
2. chapheader - the collective element containing the title, subtitle, chapnum, chapsummary, and other generic leaf elements that comprise the introduction to a chapter. Can contain any of the DTD attributes.
3. chapnum - a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.
4. chapsummary - a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.
5. endchap - if the end of a chapter contains a lot of additional information, enclose it with the endchap tag, which can contain the para element, and of the block elements, or DTD attributes.

6. attrib - a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.
7. caption - a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.
8. song - self-explanatory, can contain PCDATA, title, subtitle, verse, line, note, footnote, any of the inline elements or the DTD attributes.
9. line - a leaf element defined as a possible child element of several other elements, can contain PCDATA, any of the inline elements or DTD attributes.
10. para - can be used almost anywhere within this DTD to define a block of text, a paragraph; can contain PCDATA or the inline elements, and any of the DTD attributes.
11. simplesect - a "simple section," likely a visually blocked but small section of text, can contain one title, which must appear first, and the following: subtitle, para, poem, song or block elements, DTD attributes.
12. sect1 – “section 1,” included as a child to several elements including the preface, introduction, chapter, appendix, the sect1, sect 2, etc. are available for sections that appear almost as an outline, each section being a subsection of its parent; can contain one title which must appear first, and sect2, simplesect, para, poem, song or the block elements, and the DTD attributes.
13. sect2 - child only to sect1, can contain one title which must appear first, sect3, simplesect, subtitle, para, poem, song or block elements, and any of the DTD attributes.

14. sect3 - child only to sect2, can contain one title which must appear first, sect4, simplesect, subtitle, para, poem, song or block elements, and any of the DTD attributes.
15. sect4 - child only to sect3, can contain one title which must appear first, simplesect, subtitle, para, poem, song or block elements, and any of the DTD attributes.
16. page - a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.

Play Elements

1. speech - can contain PCDATA, speaker, stagedir, song, poem, line and the inline elements, or DTD attributes.
2. scene - usually the scene of a play, can contain the following elements: speech, poem, song, scndesc, stagedir, prose, note or the DTD attributes.
3. speaker - a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.
4. scndesc - "scene description," a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.
5. stagedir - "stage direction," a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.
6. pgroup - a grouping of persona, for example, "the chorus," can include PCDATA, title, persona, para, note or the inline elements, as well as the DTD attributes.
7. persona - a character in a story or play, can contain PCDATA, or the actor, actress and inline elements, and the DTD attributes.

8. actor - a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.
9. actress - a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.

Back Matter Elements

1. index - an index in a traditional print book contains a list of terms with the page numbers in which they can be found within the book. While they are often omitted in electronic texts, if present can include title, item, list, deflist and note elements, and can contain the following types: contents, authors, firstlines, tables, or other.
2. glossary - a list of terms with definitions, can contain: title, item, list, deflist, or note elements and any of the DTD attributes.
3. biblio - meaning bibliography, usually contains external references to relevant, related reading material. Can contain title, item, list, deflist, note, and any of the DTD attributes.
4. appendix - most appendices resemble additional chapters to a text; can contain a title or chapheader, para, poem, song, sect1, simplesect, any of the block, and the endchap elements, as well as the DTD attributes.
5. colophon - a colophon is an embellishment upon the preceding text, or can contain a brief description of how the text was produced; it can contain PCDATA, the DTD attributes, and the following elements: para, poem song, and the inline elements.

The Block Elements

1. blockquote - a long quotation included in a text, can contain: title, para, poembody, song, attrib.
2. footnote - initially declared within the block class, but also defined as a child of several other elements including poem and song; can contain PCDATA, any of the inline elements, and the DTD attributes.
3. note - a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.
4. blockgraphic - whereas a graphic is considered an inline element, a blockgraphic, typically defined by its presentation within the document, can contain title, graphic, and caption elements.

Letter Elements

5. letter - the container element for all parts of a letter: address, to, from, salut, sig, title, subtitle, para, poembody, song, line, note.
6. address - can contain PCDATA, para, line and any of the inline elements or DTD attributes.
7. to - can contain PCDATA, para, line and any of the inline elements or DTD attributes.
8. from - can contain PCDATA, para, line and any of the inline elements or DTD attributes.
9. salut – “salutation” can contain PCDATA, para, line and any of the inline elements or DTD attributes.

10. sig – “signature” can contain PCDATA, para, line and any of the inline elements or DTD attributes.

List Elements

11. list - the container element for any type of list, can contain title, list, item elements and the DTD attributes.
12. item - a very generic element that can be used for any item in a list, can contain PCDATA, para, poembody, song, simplesect and any of the block or inline elements and the DTD attributes.
13. deflist - similar to list, with the addition of “definitions;” can contain title, item, desc, def and the DTD attributes.
14. desc - a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.
15. def – “definition,” a leaf element, can contain PCDATA, any of the inline elements or DTD attributes.

Table Elements

16. table - a container element for the elements of a table, can contain title, row, caption and the DTD attributes.
17. row - a child only to table, can contain cells and have any of the DTD attributes.
18. cell - a child only to row, can almost anything: PCDATA, and any of the block or inline elements and have any of the DTD attributes.

Inline Elements

1. quote - a leaf element, can contain PCDATA, any of the other inline elements or DTD attributes.

2. emph – “emphasis,” a leaf element, can contain PCDATA, any of the other inline elements or DTD attributes.
3. ital – “italics,” a leaf element, can contain PCDATA, any of the other inline elements or DTD attributes.
4. reference - a leaf element, can contain PCDATA, any of the other inline elements or DTD attributes.
5. date - a leaf element, can contain PCDATA, any of the other inline elements or DTD attributes.
6. place - a leaf element, can contain PCDATA, any of the other inline elements or DTD attributes.
7. name - a leaf element, can contain PCDATA, any of the other inline elements or DTD attributes.
8. graphic - this inline element is actually EMPTY, but must have the required attributes desc (description) and href (location of the graphic file), and well as any of the DTD attributes.
9. txterr - an interesting inline element provided in the HTML Writer's Guild DTDs, "for use to enclose the text you think is in error optional explanation if error is not obvious." Can contain PCDATA, and an attribute, "explain," as well as any of the DTD attributes.
10. mkuperr - similar to txterr, use to enclose any markup errors, an explanation is required.
11. misc - a leaf element, can contain PCDATA, any of the other inline elements or DTD attributes.