Why use DITA to produce HTML deliverables?
Overcoming the limitations of HTML

John Hunt
Don Day
Erik Hennum
Michael Priestley
Dave Schell

September 28, 2005

The Darwin Information Typing Architecture (DITA) is an XML-based format for structuring and authoring technical content. This article explores advantages DITA provides for producing HTML content -- including easy global changes, portability through standards, superior linking and Web management, conditional processing, content and design reuse, and better writing through focused content. DITA consolidates all of the benefits in a consistent, overall information architecture that can evolve and grow along with your product information needs and delivery modes, and with the evolution of standard tools for delivering XML as the presentation mechanism.

Many of you are already familiar with the advantages of XML in general and DITA in particular for creating technical content. (If you don't know about DITA, see the introduction article.) Nonetheless, we frequently receive the same questions:

"Since my output deliverables are exclusively HTML and will stay that way for the foreseeable future, why go the route of creating content in an intermediary XML format and generating HTML? Why not produce HTML directly with an HTML editor? Why go through the extra cycles involved in producing and managing DITA XML content when it seems so much easier to simply write and produce the HTML directly, with tools that were specifically created to support doing that?"

When is it appropriate to author in HTML?
First, it's important to say yes, in some situations it might make more sense to create outputs directly in HTML, as opposed to DITA XML.

You're creating a special-purpose HTML-based UI
HTML is more appropriate than DITA when you need to create a one-of-a-kind interface, such as a calculator tool. DITA is not a replacement for the complete range of programmatic freedom afforded by HTML + JavaScript + CGI/ASP back-end code + applets + CSS + DHTML + (on and
You need to use the tool that's appropriate for the specific requirement. DITA represents an information model for technical product content. In this sense, DITA instructs the user about the system, whereas UI construction tools help instruct the system about the user, or adorn how the user sees the information about the system. DITA is the semantic framework for the information. In contrast, forms and style tools create the behavior and appearance for that framework. When you're providing behavior and appearance instead of content, it may make more sense to use HTML directly.

**You're working on a prototype or a one-of-a-kind deliverable**

For a one-of-a-kind project or deliverable, with no expectation of long-term maintenance, it may make sense to take the quick and dirty route of authoring technical product content directly in HTML. For example, you might choose to author an individual Web page or an occasional newsletter directly in HTML. You also may need to prototype content for delivery with a new product offering. Direct HTML authoring provides the ability to prototype with uniquely customized pages.

The following characteristics of prototypes and one-of-a-kind deliverables make them suitable for HTML authoring:

- It's not important that the structure and content of your information match up with similar content being produced by other writers on the same or other teams. Matching up HTML styles requires extensive up-front coordination and planning, as well as ongoing editorial vigilance.
- Consistency from page to page is not important.
- You know that styles and behavior will not need to change midstream.
- You're comfortable taking on the entire responsibility for the styling, look-and-feel, and overall presentation of the content in all possible browsers and platforms in which it might appear.

**What's the problem with authoring directly in HTML?**

However, for authoring most technical content -- including help, manuals, and other user assistance -- working directly in HTML has a number of limitations and problems:

**Difficult to make global changes**

Once you've created a set of HTML pages that follow particular style and content guidelines, it's difficult and labor-intensive to make global changes. Cascading Stylesheets (CSS) may provide some help, but often the kinds of changes you require may go beyond what CSS can accomplish. And making any change to one HTML page means you must search and replace parallel changes in all the other pages you've created. Minor inconsistencies can pose major problems for such a search-and-replace approach. Slight variations in how the HTML was created can often foil the match.

**Not extensible**

You can develop HTML content that follows agreed-upon information typing standards. However, as you encounter the need for new types of information, it's often difficult to extend an HTML
design to accommodate both new information types and the legacy types. It's also difficult to enforce agreed-upon guidelines for creating information types. With HTML, you're constantly reinventing the wheel each time you adjust your information architecture.

**Difficult to determine information completeness**

HTML doesn't easily provide a systematic way to check that a set of topics includes the full set of concept, task, and reference topics needed to document a product feature. As a result, it's difficult to gauge progress, and impossible to ensure information completeness.

**Not portable or consistent**

HTML content is notoriously difficult to share across product groups, or with external business partners. Yes, anyone with a browser can display your HTML pages, but mixing and matching HTML that follows different content and presentation models quickly becomes unwieldy. Navigation, layout, headings, and general presentation style lack overall consistency as users navigate from topic to topic -- and there's really no way to do anything about it, short of reworking each information set to match the common standard. And what happens when a business partner adds another content plug-in to your information set?

**Why author in DITA XML?**

*So, enough already! HTML may have its weaknesses, but it's the devil I know. If not HTML, tell me what's so compelling about authoring in DITA XML and then generating HTML.*

**XML and information development**

XML offers a number of advantages for information development:

- It stresses the *structure* of the content, not the *form*; layout is maintained separately in most XML designs.
- It provides for greater consistency of content, and assures greater consistency of the presentation of that content on a wider variety of output devices and formats.
- It offers ways to support conditional processing, automatic linking and link checking, and a powerful reuse model.

At IBM, the move to XML for information development allows us to take advantage of several unique features of XML:

- **Open standards:** XML provides an application- and system-independent format for sharing and exchanging content, made better when sharing organizations use an agreed-upon tagging system, such as that defined in a document type definition (DTD) or other schema.
- **Separation of form from content:** XML makes it possible to present the same source content in different formats -- for example, as Web pages, printed pages, or other delivery media. Consequently, a program can transform the presentation to give an entire Web site a new style without changing the underlying content. You can isolate product branding into separate presentation files so that specific brand styles do not interfere with reusing and integrating the content.
• **Extensible and meaningful tags:** XML tags can be designed to have a specific meaning and label specific content. For example, a zip code in an address might use a tag called "zipcode"; a step in a procedure, "step." Processing systems (such as search and personalization software) can filter and format the content, targeting and delivering it to specific groups of users.

• **Consistent tools:** The reliance on open standards provides a basis from which a wide variety of tools for creating, managing, and deploying XML content can emerge.

### Authoring and processing in DITA

At IBM, a team of information development professionals has developed DITA, an XML DTD and architecture specifically for technical content. The DITA team seeks to apply all of these generic features of XML to specific advantage for creating, managing, and deploying technical content with and about our products.

DITA's unique strength for technical information stems from two key features:

• **Topic-based and modular:** DITA is an architecture for creating and delivering topic-based, modular technical information. In DITA, the core information unit is a **topic**, which describes a single task, concept, or reference item. Because DITA is topic-based, DITA content can be combined, recombined, and reused to create online help, printed books, Web-based information centers, product support portals, and many other forms of information.

• **Based on information types that can be specialized:** An **information type** defines the role of a topic. DITA includes three information types derived from the base topic type: task, concept, and reference. A **task** topic presents the step-by-step procedure for a task. Task topics answer **How do I?** questions for a specific task. **Concept** topics provide the reason behind the tasks by defining terms and explaining concepts. A **reference** topic provides specific information, such as a command, message, program option, or API. Because DITA supports task, concept, and reference topics, writers and editors can quickly determine if a new function has been completely documented. Finding task topics that are not supported by concept topics may indicate that additional writing is required.

Through DITA specialization, you can create and enforce a consistent information architecture. For example, a specialized topic used to document a C++ API includes rules that force writers to compose a set of required content, such as a return value. Users of this kind of highly structured content become familiar with the consistent structure of the information and find they can almost intuitively locate topics -- for instance, searching for "an install step for an expert" or searching for "return values in C++".

(For examples of specializations and topic-based processing with DITA information types, see **Specializing topic types in DITA**.)

DITA leverages the advantages inherent in XML and extends beyond those advantages in the following ways:

• **Easy global changes through customized transforms:** With DITA and XSLT, you can update the structure and presentation of an entire information set by applying a consistent, core transform. You can automate things like building summary tables and listing linked
topics. And because these global changes get applied during output, you can apply different sets of global changes for different output. In this way you can generate customized outputs for print versus online, or for different platform or branding requirements, without having to edit and adjust the source each time. You can quickly respond to customer demands for new and updated product information.

- **Portable through standards:** Using DITA, product groups and external business partners can easily share and exchange content. Third parties can use common transformation and presentation models with DITA, or create specialized processing to offer views and presentation of content that is company- or brand-specific, or to transform content for reuse between DITA and other XML formats. This content portability is critical for maintaining arrangements with third-party partners and for ensuring that a writing team remains productive through business reorganizations, mergers, acquisitions, and spin-offs.

- **Linking and Web management:** DITA makes it possible to create and maintain cross-topic links from outside the topic itself. You can apply different sets of links in different situations. For example, when your topics are included in product A, the appropriate links for that product are included. For product B, a second set of links are included. Similarly, when you're incorporating content produced by another team, you can add appropriate links to their topics during processing without editing their source. You can even add links after topics have shipped to translation.

- **Conditional processing:** With DITA, you can tag parts of a topic by product, audience, or other characteristics. You can then include, exclude, or otherwise flag that content for reuse or specialized presentation.

- **Reuse:** You can reuse topics in different collections using maps, and you can reuse content between topics as well, maintaining common elements like definitions, warnings, and product names in a central place. With DITA, writers can assemble topics about a specific set of issues and publish them as a unique on-demand deliverable. For example, a customer support team might compile from existing, diverse sources a particular set of topics (such as server load and performance) that provide a customized solution to a problem reported by a major customer.

- **Focused content and better writing:** Topic-based authoring produces better writing. Categorizing content into concept, task, and reference topics ensures that users can perform tasks faster because the information is focused. Delivery tools that handle metadata can enable users to search for information based on their company role, their job responsibilities, and their task goals.

**Bottom line**

It's possible to achieve some of the above benefits through highly disciplined authoring of HTML and subsequent processing of the authored HTML. However, this quickly becomes a bits-and-pieces process. For example, you might tweak HTML to support a form of conditional processing, but in so doing make it difficult to generate a customized presentation. Then, when you tweak the HTML to improve the presentation, you might need to re-work the content and form of the topic navigation links.

XML and DITA overcome this bits-and-pieces problem of HTML. DITA consolidates all of the benefits in a consistent, overall information architecture that can evolve and grow along with
your product information needs and delivery modes, as well as the evolution of standard tools for delivering XML as the presentation mechanism.

**Acknowledgements**
This paper was developed by the DITA architects team. John Hunt, the primary author, led the work effort developing the paper.

© Copyright IBM Corporation 2005
**Trademarks**
(www.ibm.com/developerworks/ibm/trademarks/)