*Content providers are transforming their search offerings into applications tailored to specific audiences, delivering information products based on users' roles, activities, and work processes.*

**Stephen Buxton**

# Beyond Search: Content Applications

Content providers and publishers are finding more and more that they don't compete on content, but on what they can do with that content. "Content was king" back in 1997, according to Thomson Corp. Senior Vice President David Turner, in his 2006 National Federation of Abstracting and Information Services presentation ("The Thomson Transformation: Remaking a Global 500 Company," http://www.nfais.org/TurnerNFAIS06.ppt). Now content providers and publishers are moving "up the value pyramid" by competing on applications built on top of a technology platform that sits on top of content. We call these *content applications*.

People often associate *content* with a search engine and *applications* with data (and therefore a relational database management system). But if you were designing a platform for content applications today from scratch, you would not use either a search engine or an RDBMS as the starting point. You would start with a representation of semistructured content—stored and indexed efficiently and securely in a content repository. You would also need a powerful, content-focused query/programming language. In addition, you would take the learnings from the world of search engines and the world of databases, add modern content technologies such as XML and XQuery, and come up with a brand new platform.

## CONTENT APPLICATIONS

The Oxford African American Studies Center is an example of a content application delivered on a subscription-based Web site. Oxford University Press built the site, which bills itself as "the first online resource center to fully document the field of African American Studies and Africana Studies" (http://www.oup.com/online/africanamerican).

As you can see in Figure 1, this site includes the familiar keyword search box. Here, I typed in the phrase "Maya Angelou." But the application differs substantially from a typical search engine. First, the application pulls together rich, authoritative content from thousands of books, articles, and original source documents—all authored for other purposes—in various formats.

Second, the application shows the powerful user experience that can be built using rich content and underlying content technologies. An application can build, on the fly, such things as

- an at-a-glance summary of Maya Angelou, including her picture, place and date of birth, and professions;
- summaries of the top five articles on Maya Angelou;
- a time line of milestones in Maya Angelou's life, possibly correlated with other time lines (such as the American civil liberties movement);
- recommendations, showing related people or articles or sources; and
- drill-down in several dimensions—for example, by types of information (biographies, images, charts and tables, and so on) or by category (history, politics, and so forth).

1520-9202/07/$25.00 © 2007 IEEE Published by the IEEE Computer Society



Authorized licensed use limited to: IEEE Xplore. Downloaded on April 29, 2009 at 08:34 from IEEE Xplore. Restrictions apply.

**Figure 1. Oxford African American Studies Center, by Oxford University Press, is subscription based.**


None of these pages (or panels) exist in this form in the content—the application finds pieces of information, retrieves them, and composes the page on the fly.

Instead of installing a search engine that indexes all their documents and adding features the search engine provides, the folks at Oxford University Press took a different approach. Their application uses the keywords you type in as a hint and delivers meaningful information about Maya Angelou, along with a plethora of research paths to take you further. The Oxford African American Studies Center is not a search tool, it's a research tool. The site puts useful information into a context appropriate to the researcher—in short, it supplies answers, not links.

The Oxford African American Studies Center, however, still looks a bit like a search engine. Let's look beyond search and research and take a quick look at two content applications in the wild that don't act at all like search engines.

## SafariU

SafariU (http://www.safariu.com) is a joint venture between O'Reilly Media and the Pearson Technology Group. A bit like iTunes for textbooks, SafariU lets professors rip, mix, and burn their way to custom textbooks for courses. They start by searching an extensive collection of books and articles for sections, chapters, or even whole books that are most relevant to their course. Then, professors can create a mashup of the results, adding their own materials and notes. Once they have the end product just the way they want it, they can preview a low-resolution PDF of the new book (complete with a custom table of contents and an accurate, automatic back-of-book index) and order printed copies of the book to be delivered to the university store. This content application starts with search, but produces so much more—in this case, a single printed book that's both up-to-the-minute and perfectly tuned to the needs of the course. For students, each section of the book acts as a launch pad into the more detailed information found in the source reference available to them on the Web.

## PathConsult

PathConsult from Elsevier offers tools for pathologists performing differential diagnoses (http://www.pathconsultddx.com/pathCon/home). Going far beyond simple search, PathConsult can become an integral part of the diagnostic process, helping pathologists perform diagnoses more quickly and accurately. For example, a pathologist might select two or more diagnoses to see a side-by-side comparison with "diagnostic pearls" (see Figure 2), show cytology images and descriptions, then show side-by-side stains comparisons with related information (such as stain positivity). At any step, additional information about any aspect of a diagnosis is just a click away. This application is based on pictures (x-rays, microscope slides, and so on) and text (information about diseases and diagnostic procedures) drawn from standard medical texts. But PathConsult, developed in consultation with practicing pathologists, brings the reference material to life by dynamically placing it into the context of the diagnostic process specific to each patient, giving pathologists the information they require as they

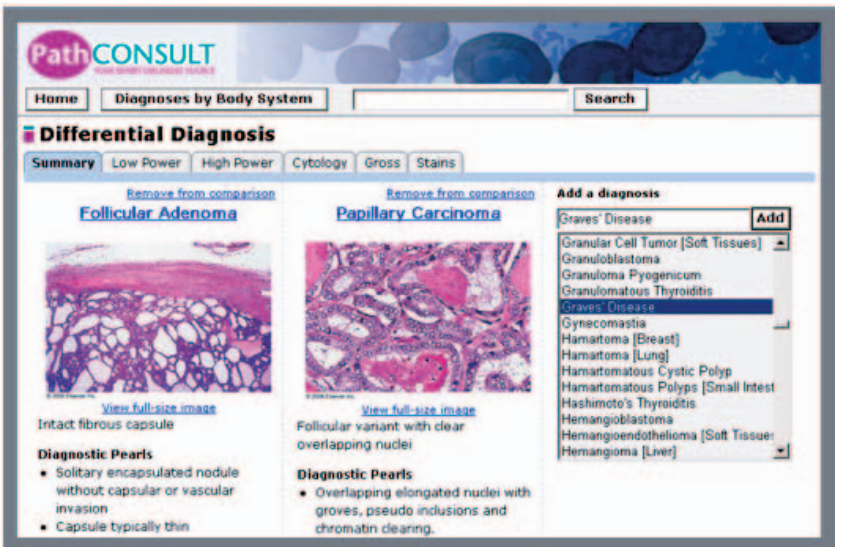


Authorized licensed use limited to: IEEE Xplore. Downloaded on April 29, 2009 at 08:34 from IEEE Xplore. Restrictions apply.

need it in the course of performing a truly mission-critical activity.

Content applications, then, can incorporate the role, activity, and process of the end user into an interactive, personalized information product. Successful content applications simplify and accelerate the user's task by contextualizing content into the user's specific situation.

**Figure 2. Elsevier's PathConsult is a diagnostic comparison tool.**




## CONTENT

In this article, when I talk about content I mean information, largely textual—documents, if you will. Content could be anything from books to journal articles to emails to technical repair manuals. The information in content is conveyed not only by words, but also by its structure and, in many cases, metadata.

If content structure were fixed and regular, we might be tempted to handle that content in an RDBMS. But content structure is different from, say, the structure of a purchase order—it's generally much more complex, deeply nested, and more flexible. Content structure varies according to its source and changes over time (as providers change their representation or enrich their content).

Content applications are built on these basic operations:

- *Edit*. If the content is stored in documents (in blobs of text), you must always update a whole document. You should be able to update any part of a document—a single chapter, paragraph, figure, table—without changing or locking any other part.
- *Store*. In many enterprises, content is scattered across file systems, databases, and Web servers. But if you store content in an intelligent repository you can better manage, secure, and make use of it. The repository should store content in a format that reflects its structure, so that you can search, retrieve, and update any part of any document quickly.
- *Enrich*. Content can be enriched in a number of ways. You can add metadata, add structure (such as calling out code samples, or emergency procedures), call out facts in flowing text (names of people, places, companies), and add tags, links, and ratings for personalization. Enriching content builds value, and generally happens over time, manually and automatically.
- *Find*. Document-level search is not enough. You must be able to do fine-grained search and retrieval, so you can make complex requests like "Show me the first paragraph of any section (in any document) whose title contains 'Maya Angelou', plus the source and category for each document, plus the alt-text and URI of any image with 'Maya Angelou' in the caption or in the subsection title directly above it."
- *Reuse/repurpose*. You need to be able to reuse and repurpose any-sized chunks of text. For example, you can deliver many variants of an aircraft repair manual by reusing the parts that are common to all airlines and including airline-specific instructions and procedures where appropriate.
- *Deliver/publish anywhere*. Whether the output of your content application is a custom book or a press view of an article, you want to be able to publish it anywhere, in any format—a Web page, a PDF document, a result from a Web service or RSS feed, or a cell phone message. You also want to publish it in innovative, interesting ways.
- *Analyze/mine/discover*. Analyzing and mining large bodies of content can produce tremendous value. For example, you might discover what's hot by asking "What words and phrases are emerging among new medical articles?" or "Show me a tag cloud for the abstracts of computer articles that mentioned Linux this week." (See Figure 3 for an example of analyzing content.)

If you could do all that, you could build powerful content applications and compete on the richness of your content as well as on the rich user experiences you could offer on top of that content.



Authorized licensed use limited to: IEEE Xplore. Downloaded on April 29, 2009 at 08:34 from IEEE Xplore. Restrictions apply.

**Figure 3. O'Reilly labs' content statistics for Jason Hunter's *Java Servlet Programming, 2nd edition*.**


(SGML), which was designed for marking up content such as technical manuals. With XML, you can represent arbitrarily deep and complex structures (not just simple data structures such as purchase orders). You can add new tags and new information wherever you want. You can tie down the structure up front, but you don't have to—you can enforce as much or as little conformance to a structural template (an XML Schema) as you choose.

*XML content* yields some unique requirements for storing, managing, and searching XML:

- *Schema-agnostic*. XML content doesn't lend itself to tight Schema control. Typically, you want to be able to store any XML content, no matter what the structure, and manipulate it over time so that the most often-used parts of the structure line up. And you don't want these changes to slow down subsequent retrieval or manipulation.
- *Universal indexing*. If you index all the words and all the structure, you can get great performance right out of the box without knowing anything about the content up front.
- *Expressive, content-focused programming language*. You need a tool built for the job of expressing all those complex questions and manipulating the content and structure together.

## CORE TECHNOLOGIES FOR BUILDING CONTENT APPLICATIONS

Two core technologies let you build great content applications: XML and XQuery.

### Content and structure: XML

All content has some natural structure. For example, we can describe Shakespeare's works in plays, acts, scenes, speeches, and lines. Structures for books and articles include chapters, sections, subsections, and paragraphs. Similarly, subparagraph structure includes tables, figures, captions, bulleted lists, and so on. XML can represent not only natural structure but also metadata—author name, privacy level, target audience—as part of the content. (Shakespeare's plays, marked up in XML, are available at http://www.ibiblio.org/xml/ examples/shakespeare/.)

Extensible Markup Language (XML) started life as a subset of Standard Generalized Markup Language

With universal indexing, for example, you can ask arbitrarily complex questions across all the content. You don't need to reload or reindex when you discover a new question or when you enrich or restructure the content. This leads to extremely agile development of applications over heterogeneous content. You can get value from your content application almost immediately and improve both the content and the application incrementally over time.

### Querying and manipulating content with structure: XQuery

XQuery is a language defined by the World Wide Web Consortium (W3C) to query and manipulate content and structure. Think of it (though only loosely) as SQL for XML—where SQL lets you query and manipulate data



Authorized licensed use limited to: IEEE Xplore. Downloaded on April 29, 2009 at 08:34 from IEEE Xplore. Restrictions apply.

in tables (rows and columns), XQuery lets you query and manipulate the content and structure of XML documents.

Since XQuery includes XPath, let's start with XPath. XPath lets you pull out a particular piece of an XML document, using "/" and "//" to represent "child" and "decendant," and "[]" to enclose a predicate.

For example, the path expression

```
/PLAYS/PLAY[TITLE="The
Life of Henry the
 Fifth"]//PERSONA
```

returns a sequence of all the persona anywhere in the play with the title *The Life of Henry the Fifth*. More simply, it answers the question, Who are the characters in *The Life of Henry the Fifth*?

XPath is useful, but what's needed is a more powerful set of expressions. XQuery adds the FLWOR expression (generally pronounced "flower"). The FLWOR expression lets us walk the XML structure tree (with XPath) and also iterate over the results, doing something useful with each one (see Figure 4).

**Figure 4. An XQuery to list each character in *The Life of Henry the Fifth*, along with his or her first line.**

```
for $x in input()/PLAY[TITLE="The Life of Henry
    the Fifth"]//PERSONA
let $printable-name := lower-case( $x )
let $speeches := input()/PLAY[TITLE="The Life
    of Henry the Fifth"]//SPEECH[SPEAKER=$x]
order by $printable-name ascending
return
    <li>{ $printable-name }<br/>
    { if ( $speeches )
    then $speeches[1]/LINE[1]/ text()
    else "non-speaking part"
    }
    </li>
```

In just the few lines shown in Figure 4, I collected all the characters from a particular play (`for`). The XPath in the for clause can be arbitrarily complex, and it can include arbitrary predicates. In this case, I iterated over the characters in the play *The Life of Henry the Fifth*.

In addition, I assigned a couple of iteration variables (`let`). `$printable-name` gives us a convenient way of talking about the character's name, converted to lower case. `$speeches` is assigned the result of another XPath (it could have been a whole FLWOR), which pulls together all the speeches that character makes.

The code in Figure 4 also let me arrange the results in some order (`order by`). I chose ascending alphabetical order of `printable-name`. Finally, the code returned the result from each iteration, as a sequence of results (`return`). I chose to return the printable name of the character, followed by the first line of the first speech he makes.

This is a simple example, but notice how powerful the FLWOR expression is. The result is not just a list of speeches, but a list of characters in the play plus the first line from their first speeches. We could order this by the character's name or by the number of lines spoken. We could pull in speeches that the same character made in other plays, and we could classify each character according to his or her longest speech in any play. This is the power of using XML and a full-blown query language, rather than unstructured content and basic keyword search.

Notice, too, that the XQuery may contain HTML tags (and HTML tags may contain XQuery expressions). You can paste the example XQuery into an XQuery-aware Web page to display the results as a list. In the same way, XQuery expressions can appear inside (or outside) of any XML tags, so that the results could be a SOAP message or an RSS feed.

## XQuery for content

Now that you've seen how basic XQuery works, let's consider how we need to extend basic XQuery for content.

First, we need full-text search capability. The marriage of full-text search and XQuery is tremendously powerful. It enables arbitrarily fine-grained full-text search and retrieval over arbitrarily structured content, and it's the foundation of content applications.

The W3C is discussing XQuery Full-Text extensions (http://www. w3.org/XML/Query/#specs), but some vendors have already implemented their own extensions. Figure 5 shows how XQuery Full-Text might work (I've used Mark Logic Corp.'s syntax to illustrate the principle, because I can easily test the example).

This example uses the extension functions `cts:search()` and `cts:contains()` for full-text search. Figure 5 only shows simple keyword search, but of course you could search for phrases, stems, synonyms, and so on.

In Figure 5, I first identified all the plays that contain the word "battle," then I filtered them down to only the ones that have a speech that contains the word "castle." Then, for each castle speech in each war play, I returned the speaker, the first line of the castle speech, and the actual line that contains the word "castle." I also ordered the plays

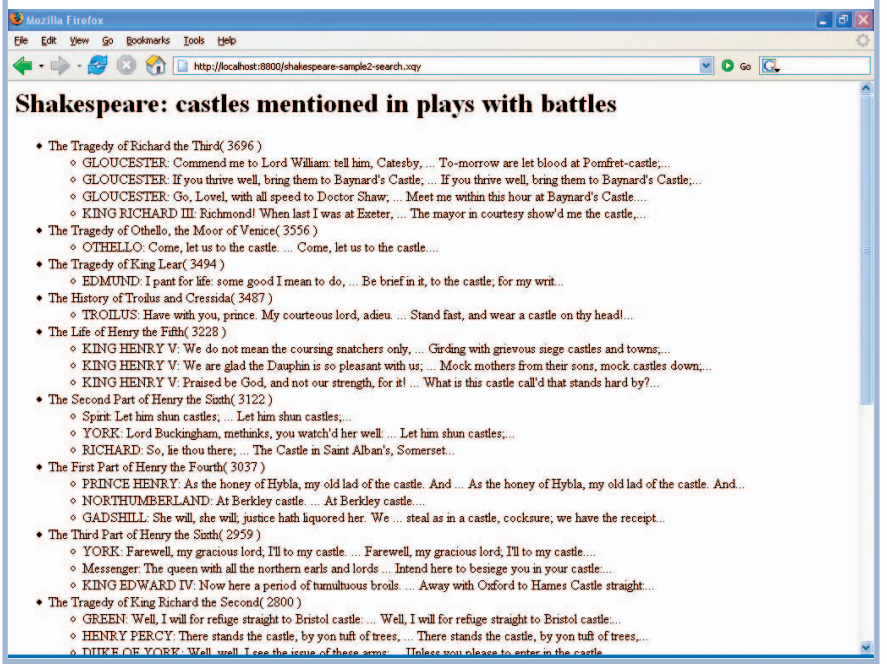


Authorized licensed use limited to: IEEE Xplore. Downloaded on April 29, 2009 at 08:34 from IEEE Xplore. Restrictions apply.

**Figure 5. An XQuery to show lines about castles (and their speakers) in plays with battles. Figure 6 shows the results.**

```
<h1>Shakespeare: castles mentioned in plays with battles</h1>
<ul>{
for $warPlay in cts:search( input()/PLAY, "battle")
where cts:contains
( $warPlay//SPEECH, "castle" )
order by count( $warPlay//LINE ) descending return <li>{ (
$warPlay/TITLE/text(), "( ", count(
   $warPlay//LINE ), " )" ) }
   <ul>{
     for $castleSpeech in $warPlay//SPEECH where cts:contains( $castleSpeech,
     "castle" )
     return
       <li>{ (
         $castleSpeech/SPEAKER/text(),
         ": ",
         $castleSpeech//LINE[1],
         " ... ",
         $castleSpeech//LINE[ cts:contains( ., "castle" ) ][1],
         "..."
         )
       }</li>
     }</ul>
   </li>
}</ul>
```

**Figure 6. Results of the XQuery in Figure 5.**




by number of lines and showed the number of lines by the play title. Figure 6 shows the results.

XQuery, with extensions for full-text search (like the full-text extensions just described) and extensions for updates (which I don't have room to discuss here), is the foundation for content applications. You could use it to write a familiar-looking search page, with a box for document-level keyword search and a list of links to documents in the results. But that's just a trivial example—XQuery can do much, much more.

The ability to import modules means you can write and reuse libraries of XQuery code (see http://developer.marklogic.com/ for some examples). This makes XQuery suitable for building entire applications, not just queries. Proven programming features such



Authorized licensed use limited to: IEEE Xplore. Downloaded on April 29, 2009 at 08:34 from IEEE Xplore. Restrictions apply.

as try/catch blocks (for easy error handling) can also be added to XQuery as extensions. XQuery for content also needs to support granular inserts, deletes, and updates transactionally. You'd also want to be able to write triggers linked to your content.

Together, these extensions give you the control and consistency over content that RDBMS users have come to expect for data.

The promise of content applications is that content providers and publishers can write real applications against content, to help a known body of users, with known roles, complete a set of tasks with extraordinary precision and efficiency.

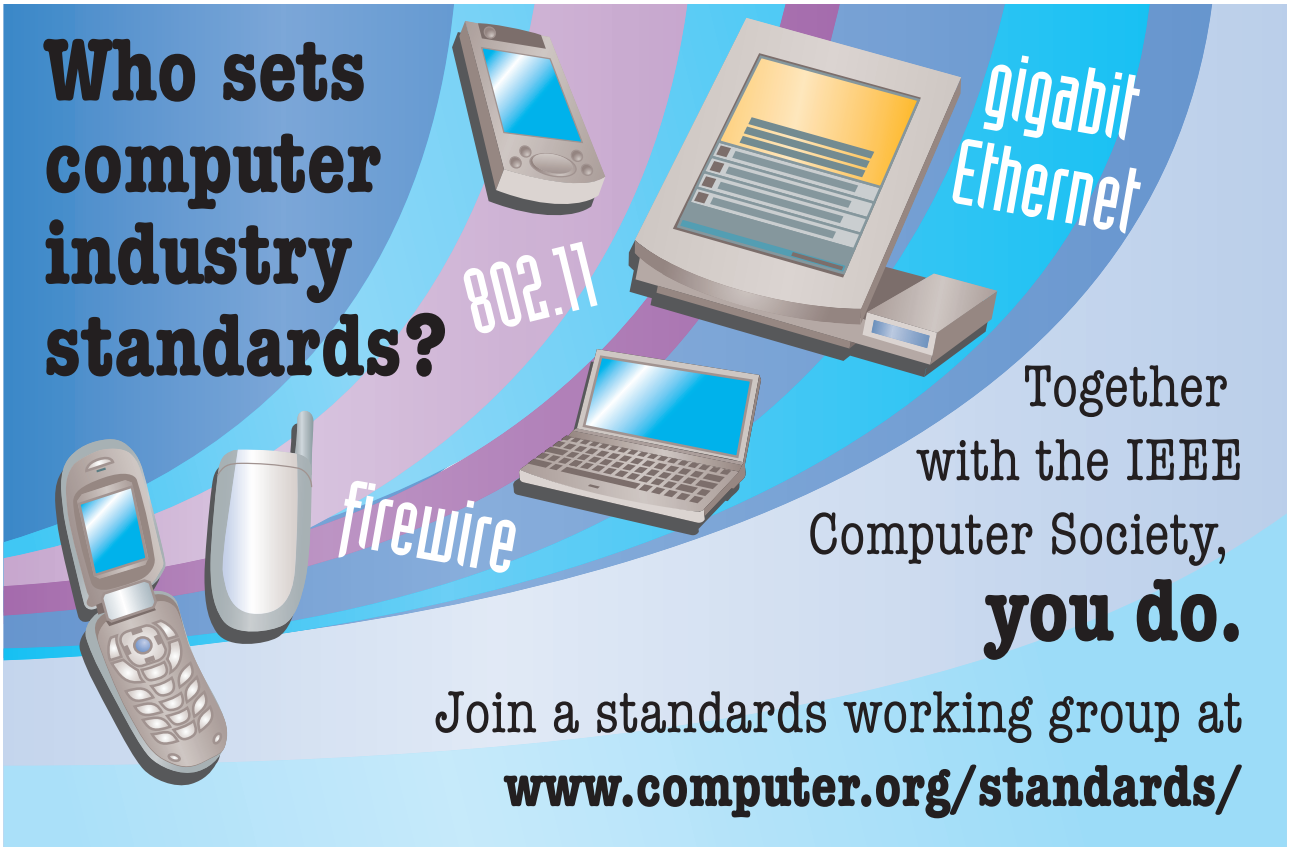
The world is changing into a place where content is essentially freely available to everyone. In this new world, content providers must differentiate themselves on the rich applications they provide for their users. The winner will be the content provider who best helps users accomplish their tasks. Take another look at the Oxford University Press African American Studies site. Wouldn't you pay a subscription fee for this kind of service, even if you could get the raw content (say, as a stack of books)for free?

Content providers and publishers are just the early adopters of this new wave of content applications. Content applications will become mainstream as enterprise applications also evolve to manage, search, manipulate, and present information in innovative ways.

So, the future of search entails rethinking what content is, and what we can do with it. With modern core technologies—XML, XQuery, and full-text search—it's possible to deliver sophisticated content applications that were impossible or impractical before. These new technologies, and XML content servers that capitalize on them, will spur a new wave of content applications that change the way we think about content, much as RDBMSs changed the way we thought about data. XML content servers will become ubiquitous, and the now-familiar basic keyword search page will become a thing of the past. ■

***Stephen Buxton*** *is the director of product management for Mark Logic Corp. (http://marklogic.com). He coauthored (with Jim Melton) the book Querying XML: XQuery, XPath, and SQL/XML in Context. Contact him at stephen.buxton@marklogic.com.*

For further information on this or any other computing topic, please visit our Digital Library at http://www.computer.org/publications/dlib.






Authorized licensed use limited to: IEEE Xplore. Downloaded on April 29, 2009 at 08:34 from IEEE Xplore. Restrictions apply.