

XML JOURNAL

THE ULTIMATE XML ENTERPRISE RESOURCE

June 2003 Volume: 4 Issue:6

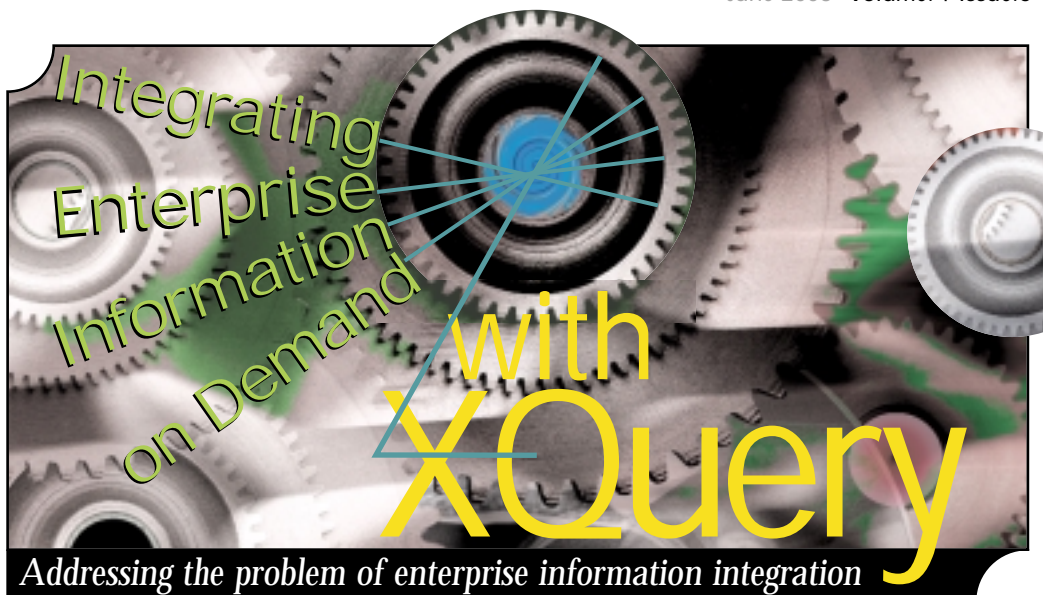
xml-journal.com

From the Editor
Beyond Integration
by Hitesh Seth pg. 3

Industry Commentary
Leveraging XML Open
Standards to Integrate
Your Business
by Rachel Helm pg. 5

Certification
XML Certification Quizzer:
Review of Key Concepts
by Joel Amoussou pg. 30

XML News
XHTML 2.0 Working
Draft Published
...
XML Key Management
Requirements Published
pg. 34



 **Feature: XML Authoring in the Financial Services Industry** *MFS Investment Management and Moody's Investors Service reap the benefits*

Max Dunn



6

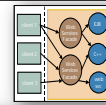
 **Integration: Enabling Integration of Internal Data and External Business Information** *New technologies deliver accurate snapshots of customers, prospects, and competitors*

Patti Purcell

12

 **Feature: Applying Design Patterns to Web Services Architectures** *A few well-known patterns can simplify your development efforts*

Chris Peltz



14

 **Feature: Introducing Microsoft InfoPath 2003** *The key to reducing your 'mountain of paper'*

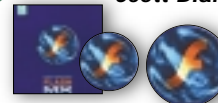
Thom Robbins



18

 **XML & Flash: Using XML & Flash to Create a Server-Agnostic Application** *XML is an important component of this ambitious project*

Scott Blanchard



22

 **Feature: Integrating Enterprise Information on Demand with XQuery** *Addressing the problem of enterprise information integration*

Michael Carey,

Daniela Florescu,

& Nitin Mangtani 24



International Web Services Conference & Expo **WEST**
Web Services Edge 2003
SEPT. 30 - OCT. 2, 2003
SANTA CLARA, CA

XML EDGE
conference & expo

EXTENDING THE ENTERPRISE
WITH WEB SERVICES THROUGH JAVA,
J2EE, WEBSERVICE, SOAP, XML
AND XML TECHNOLOGIES

Focus on XML

WebSphere

COMING Sept. 30-Oct. 2



SYS-CON MEDIA

Macromedia

www.macromedia.com/go/cfmxad

FOUNDING EDITOR

Ajit Sagar ajit@sys-con.com

EDITORIAL ADVISORY BOARD

Graham Glass graham@themindelectric.comCoco Jaenicke cjaenicke@attbi.comSean McGrath sean.mcgrath@propylon.comSimeon Simeonov talktosim@polarisventures.com

EDITORIAL

Editor-in-Chief

Hitesh Seth hitesh@sys-con.com

Editorial Director

Jeremy Geelan jeremy@sys-con.com

Managing Editor

Jennifer Van Winckel jennifer@sys-con.com

Editor

Nancy Valentine nancy@sys-con.com

Associate Editors

John Evdemon jevdemon@sys-con.comJamie Matusow jamie@sys-con.comGail Schultz gail@sys-con.comJean Cassidy jean@sys-con.com

PRODUCTION

Production Consultant

Jim Morgan jim@sys-con.com

Art Director

Alex Botero alex@sys-con.com

Associate Art Directors

Louis F. Cuffari louis@sys-con.comRichard Silverberg richards@sys-con.com

Assistant Art Director

Tami Beatty tami@sys-con.com

CONTRIBUTORS TO THIS ISSUE

Joel Amoussou, Scott Blanchard, Michael Carey,

Max Dunn, Daniela Florescu, Rachel Helm,

Nitin Mangtani, Chris Peltz, Patti Purcell,

Thom Robbins, Hitesh Seth

EDITORIAL OFFICES

SYS-CON MEDIA

135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645

TELEPHONE: 201 802-3000 FAX: 201 782-9637

XML-JOURNAL (ISSN# 1534-9780)

is published monthly (12 times a year)

by SYS-CON Publications, Inc.

Periodicals postage pending

Montvale, NJ 07645 and additional mailing offices.

POSTMASTER: Send address changes to:

XML-JOURNAL, SYS-CON Publications, Inc.,

135 Chestnut Ridge Road, Montvale, NJ 07645.

©COPYRIGHT

Copyright © 2003 by SYS-CON Publications, Inc. All rights reserved.

No part of this publication may be reproduced or transmitted

in any form or by any means, electronic or mechanical,

including photocopy or any information storage and retrieval

system, without written permission. For promotional reprints,

contact reprint coordinator. SYS-CON Publications, Inc.,

reserves the right to revise, republish and authorize its readers

to use the articles submitted for publication.

All brand and product names used on these pages

are trade names, service marks, or trademarks of their respective

companies. SYS-CON Publications, Inc., is not affiliated

with the companies or products covered in XML-Journal.



Beyond Integration

WRITTEN BY HITESH SETH



Even though interoperability and making systems and enterprises work together have been the main goals of XML and Web services technologies and standards, the quest for a better way to develop applications has led XML in another direction: around consuming XML/Web services to developing modular and dynamic user interfaces.

Standards like XForms, Web Services for Remote Portals, Web Services User Interface, and a whole slew of implementations – along with products such as Microsoft's recent addition to the Office Suite, InfoPath – take this approach further and allow business analysts to become the next-generation developers, or rather assemblers of dynamic applications.

If you are a member of the IT division of your company, I'm sure you've been bombarded with requests for "micro-applications" such as phone books, contact lists, lists of assets, and other such applications that, when drilled down into technical details, involve a simple mechanism for capturing and retrieving structured data in a user-friendly way. As technologists we have often taken these requirements and built either generic "e-forms" solutions or a custom specific solution, depending on our expertise level as well as the time allocated for doing the work. While implementing such micro-applications, we have often contemplated a simpler, more extensible, and better way to do this.

Enter XML, providing a portable document format that tags the data stored in it. Top it off with XML Schema, and we have defined the various error-checking mechanisms in the document. Web services are making a huge splash in the industry as we realize that they can be used as a portable, standard way of moving these "structured" tagged/typed documents around. The missing piece of the puzzle is an easy-to-use interface with which technology-savvy business analysts (rather than developers) can build their own micro-applications. InfoPath fits very nicely into this category. Although there have been a number of products/initiatives around this, Microsoft has truly captured the essence of usability to create a killer application. Although I don't like *everything* about InfoPath (for instance, we need a Web-based viewer/editor/control for InfoPath as well as alignment of the functionality with standards initiatives such as XForms), I truly consider it one of the best

applications of XML and Web services. For more details, look no further – this issue includes a great introductory piece on InfoPath by Thom Robbins.

Another key development regarding e-forms is major announcements from Adobe regarding extensive support of XML in the Acrobat product line. With millions of Acrobat Readers installed on computers and devices (and available for a number of PDA platforms as well), PDF has definitely earned respect as a true portable document format. Mix XML with PDF, and what you get is a rich, structured, portable, and extensible document format. If you've ever interacted with a government agency, chances are that you've filled out a number of forms. Many government agencies have actually created forms that can be filled out and printed using Acrobat Reader. I can well imagine the day when government agencies will create XML-based forms defined using a schema and rendered through Acrobat Reader.

Going further, integration has brought our attention to real-time or batch transactional data integration from one system or partner to another. We're also seeing a number of developments around information and semantic integrations between multiple information sources. For instance, an industry-focused market trends report may be derived through mining a quarter-old data mart, watching a couple of Internet news feeds for third-party information on the source, and perhaps looking at an internal file store. Similar to the well-established EAI acronym, a new acronym – EII, which stands for Enterprise Information Integration – is coming into the limelight.

A number of key infrastructure/database vendors have introduced a slew of products around this phenomenon as well. XQuery, the upcoming W3C XML query language, is a standard that has received a lot of attention in this area as well – look for details on XQuery in the first installment of a two-part series in this issue.

Enjoy this issue of *XML-Journal*, as it takes you through the topics related to integration...and beyond. ☘

AUTHOR BIO

Hitesh Seth, editor-in-chief of XML-Journal and XML Track chair for the Web Services Edge Conference, is the chief technology officer of ikigo, Inc., a provider of business activity monitoring solutions.

HITESH@SYS-CON.COM

IBM

ibm.com/websphere/seeit



135 Chestnut Ridge Rd., Montvale, NJ 07645
TELEPHONE: 201 802-3000 FAX: 201 782-9637

PRESIDENT and CEO

Fuat A. Kircaali fuat@sys-con.com

BUSINESS DEVELOPMENT

VP, Business Development

Grisha Davida grisha@sys-con.com

ADVERTISING

Senior VP, Sales & Marketing

Carmen Gonzalez carmen@sys-con.com

VP, Sales & Marketing

Miles Silverman miles@sys-con.com

Advertising Director

Robyn Forma robyn@sys-con.com

Director of Sales & Marketing

Megan Mussa megan@sys-con.com

Advertising Sales Manager

Alisa Catalano alisa@sys-con.com

Associate Sales Managers

Carrie Gebert carrieg@sys-con.com

Kristin Kuhnle kristin@sys-con.com

SYS-CON EVENTS

President

Grisha Davida grisha@sys-con.com

Conference Manager

Michael Lynch mike@sys-con.com

CUSTOMER RELATIONS

Circulation Service Coordinators

Niki Panagopoulos niki@sys-con.com

Shelia Dickerson shelia@sys-con.com

JDJ STORE

Manager

Rachel McGouran rachel@sys-con.com

WEB SERVICES

VP, Information Systems

Robert Diamond robert@sys-con.com

Web Designers

Stephen Kilmurray stephen@sys-con.com

Christopher Croce chris@sys-con.com

Online Editor

Lin Goetz lin@sys-con.com

ACCOUNTING

Accounts Receivable

Kerri Von Achen kerri@sys-con.com

Financial Analyst

Joan LaRose joan@sys-con.com

Accounts Payable

Betty White betty@sys-con.com

SUBSCRIPTIONS

SUBSCRIBE@SYS-CON.COM

1 888 303-5282

For subscriptions and requests for bulk orders,
please send your letters to Subscription Department

Cover Price: \$6.99/issue

Domestic: \$69.99/yr (12 issues)

Canada/Mexico: \$89.99/yr

all other countries \$99.99/yr

(U.S. Banks or Money Orders)

Back issues: \$12 U.S. \$15 all others

Leveraging XML Open Standards to Integrate Your Business

WRITTEN BY RACHEL HELM

I've been focused on defining product strategy for business integration software for the past seven years. During that time I've watched XML go from being a fledgling document standard with lots of potential to a core technology that is critical for business integration. In this article, I'm going to discuss some of the reasons behind XML's meteoric rise in the business integration space and some of the ways we at IBM are leveraging XML in our integration products.

Why Has XML Become Central in Integration Products?

It's flexible: XML's ability to handle a diverse array of information formats and to readily transform data from one kind of document type to another makes it the perfect medium for integration. You can use XML for processing business documents that are simple in structure as well as for processing documents that are complex and highly structured. XML can also be used to query relational databases accessing sets of data in a rows and columns tabular format. It can then take the data, regardless of the source, structure, or format, and aggregate, transform, and ultimately render it in any format desired. This flexibility extends beyond data formatting and description. XML can be used to implement business rules as well as business process flow logic. These are all valuable characteristics for an integration solution that needs to mediate between many different data sources and destinations and must also enable business users to define business rules and logic that crosscut multiple applications and data sources.

It's pervasive and it's standards based: Perhaps the most obvious advantage is how pervasive XML has become. Today, XML is supported on just about every operating system in the marketplace. While broad availability is important to the success of most technologies, for integration, it's an absolute requirement. Integration is about hooking together disparate systems and people in a single coherent business process. It must, therefore, leverage technology that will be supported by whatever system(s) you communicate with on the other end of the wire. Open standards play a critical role in this. It's not enough that all participants support XML. All participating components in an integrated process, regardless of the vendor or developer that built the components, must understand the same vocabulary and implement the same grammar. Open standards provide us with these common vocabularies and grammars.

Business user acceptance and adoption: The driving force behind application and B2B inte-

gration is the need to optimize how you do business. Optimizing how you utilize IT resources is a secondary albeit valuable benefit. Consequently, IT and line of business users must work closely together on integration projects if projects are to succeed. Broad support for XML standards by tools for both IT users and for line of business users and XML's ability to render data and business process logic in formats suitable for both audiences have resulted in rapid adoption of XML technology by both audiences, which facilitates the cross communication successful integration projects require.

Almost all new industry-specific standards being defined for the purpose of integrating business partners are XML based. Line of business users are not only involved in defining these standards, they often are the driving force behind them. It is these standards bodies that define the common vocabularies and grammars that B2B integration relies on. Examples of some of these standards include UCCNET in the retail industry, HIPPA in insurance and health care, SWIFT and FIX in the financial services market, and RosettaNet in the technology industry. All of these standards leverage XML to some extent.

Many of IBM's customers have chosen to leverage XML in order to revitalize their EDI solutions rather than completely replace them. Why? Because they've got tremendous investment in the semantic content of those EDI transaction formats they defined with their partners. Why abandon those partner agreement formats while they are still applicable? The VANs, on the other hand, are often no longer cost effective. So our customers move their existing EDI transactions over to a TCP/IP-based platform in order to have the best of both worlds: leverage still-relevant transaction document formats along with the cheap pervasive technology of the Internet. What technology do they use as the data format bridge? You guessed it - XML.

How IBM Leverages XML

The WebSphere Business Integration technology stack nicely reflects XML's many uses in integration. Its use in our products is certainly pervasive. Our business integration products address a broad swath of integration needs including tight application-to-application integration, human workflow, and loosely coupled integration between business partners. Today, XML is the wire format most often used across our business integration technology stack. Our runtime components and all of our tooling leverage XML as a base technology for sharing infor-

~continued on page 10~



HOME



Enterprise Solutions



Content Management



Data Management



XML Labs



XML Authoring in the Financial Services Industry

WRITTEN BY
MAX DUNN

MFS Investment Management and Moody's Investors Service reap the benefits

XML is fast becoming an integral part of information management workflows in the financial services industry, and the trend is moving toward even wider adoption.

According to the analyst firm ZapThink:

- The financial services sector spent more than \$195 billion on information technology in 2001, with \$985 million invested in XML technologies in 2002.
- Expenditures on XML technologies in the financial services sector will grow to more than \$8.3 billion by 2005.
- XML-based content management and single-source publishing can reduce the total cost of publishing by up to 75%.

In the financial services industry, effective information management can make the difference between success and failure. Individual investors, for example, want fast access to the current value of their mutual funds and other investments. They want to access such information via an expanding range of output formats – some want printed documents, while others want to look up information via a Web browser, a telephone, a PDA, or another device. There are also trends toward an increasing amount of customization and personalization of content delivery.

XML helps companies meet such needs, letting them keep existing clients happy while continuing to attract new business.

In this article we will look at the general characteristics of an XML solution, as well as some specific ways that XML has helped organizations in the financial services industry manage their information. We will consider the migration to structured authoring of two companies, MFS Investment Management and Moody's Investors Service. We will consider the challenges faced by these organizations, the way XML helps solve those challenges, and the benefits that XML solutions can provide.

The Power of Structure

Because XML provides a means of defining the structure of information, content maintained in XML is guaranteed to be structurally consistent. This consistency facilitates information exchange, as the programs involved know what to expect and can verify that the messages exchanged meet those expectations. The structure of XML can also be mapped to and from other forms of structured information, such as data housed in relational databases.

Consistent structure is also a great benefit when information is published: formatting rules, maintained in templates or stylesheets, can specify how XML is delivered to specific presentation formats. Such a template-based workflow, in which source content is maintained independently of how it is ultimately presented, enables single-source publishing (often called simply

"single-source"). Instead of maintaining distinct versions of a document for each form of publication (for example, distinct HTML and page layout files), updates can be made to a single source, and changes to this XML source are automatically reflected in all forms of published output.

Organizations are moving to XML both actively and passively: while some companies model their information with XML as part of a strategic plan, others simply find that the next revision of their core software supports XML. Such support ranges from behind-the-scenes uses of XML, such as storing initialization or configuration files in XML, to support that is integral to the high-level functionality of the application, such as the new capability of most current relational databases to return query results as XML.

XML Authoring Technologies

XML authoring tools are critical to managing information with XML. Such tools share a common set of core functions, including at a minimum the ability to parse, validate, open, edit, and save XML. Different tools extend this minimal functionality in different ways, providing capabilities such as integrated formatting, "tree-view" interfaces for navigation, automatic conversion of non-XML formats to XML, and interfaces to XML-enabled content management systems. Most XML authoring tools also provide a means of programming and/or scripting the application via an API, letting users and IT departments customize the authoring environment or create a custom interface with other applications.

WYSIWYG XML authoring

Adobe FrameMaker 7.0 is an XML authoring tool particularly relevant to financial services organizations. Perhaps most important, FrameMaker deeply integrates authoring and publishing functionality, providing true WYSIWYG XML authoring capability. With FrameMaker, users can see in real time how their XML will look when published to print or Adobe PDF format. WYSIWYG authoring lets authors know as they work on a document precisely how the document will paginate when printed: authors can see, for example, whether the word they're writing wraps to another line, or whether it begins a new page. Seeing how a document will paginate can be important when print output is a concern (and print output is a fundamental requirement for most organizations in the financial services industry), and page numbers can be useful reference points in a collaborative authoring environment.

Because FrameMaker is tightly integrated with Adobe PDF, authors can also easily control features of Adobe PDF that are specific to the electronic format, such as bookmarks and hyperlinks. While the FrameMaker user interface is clearly modeled

around a print and Adobe PDF paradigm, FrameMaker also comes with Quadralay Webworks Publisher, Standard Edition, which offers built-in publishing capabilities for other output formats (see Figure 1). The fact that FrameMaker opens and saves compliant XML, of course, also makes it compatible with a growing range of other XML-based publishing tools.

Other established XML authoring tools include Arbortext Epic Editor and Corel XMetaL. Arbortext Epic provides formatting capability for print and Adobe PDF output via an optional add-on, which lets users define formats using FOSI (an older, SGML-based standard) or XSL-FO. While Epic provides GUI tools to aid in the generation of both types of stylesheet, it does not provide complete WYSIWYG authoring capability, but rather uses a distinct “screen FOSI” to render the document as it appears in the user interface. This can provide a general sense of how the document will look when rendered, but does not go so far as to provide real-time indications of the precise way the document will be formatted when printed.

XMetaL is still less print-publishing focused, offering CSS formatting within the authoring environment, but leaving most of the work of Adobe PDF generation to the user. It provides a customization option by which users can automatically format their XML with FOP (Formatting Object Processor), an open source program from the Apache Project, but leaves it up to users to develop their own stylesheets.

Guided XML authoring

Despite their different degrees of focus on print and Adobe PDF publishing, FrameMaker, Epic, and XMetaL have many common XML authoring capabilities. All provide real-time validation, a parallel tree-like view of XML with which to navigate and edit, and visual cues that tell authors which tags are valid in the current context. The StructureView of FrameMaker, for example, lets authors easily select a range of XML content, and move that content to another location (see Figure 2). As the user moves the insertion point to potential destination locations for the current selection, visual cues (for example the check mark next to the Para element in Figure 2) indicate whether or not the resulting change would “break” the structural integrity of the XML (i.e., result in the XML being out of conformance with its DTD). A similar functionality exists in both Epic and XMetaL.

Authoring tool as repository interface

Increasingly, XML content is managed in some form of database or repository. XML support of the Documentum 5 enterprise content management system, for example, lets authors “check in” or “check out” XML files, which can be entire documents or document components. Fine-grained security, such as that provided by Documentum 5, can also provide powerful support to collaborative authoring. With such a system, multiple authors can maintain distinct sections of a document, with a view of the current state of the entire document available to all. Content management systems also typically provide versioning capability, so edits can be undone as needed, and workflow management capability, such that review and approval processes are triggered automatically based on authoring activity. XML authoring tools are often customized to interface with such systems. While robust customization can be convenient, a basic form of compatibility exists by virtue of the XML support common to authoring tools and repositories.

XML authoring tools can serve a range of roles in relation to such repositories. In some workflows authoring tools are used primarily to edit source data, i.e., maintain the content residing in the repository. In other cases, they are more oriented toward managing the publishing process, formatting content that streams from the repository but not necessarily modifying or

editing the source of that content. A common workflow in the financial services industry, for example, is to pull data (in the form of XML) from a relational database, and use an authoring tool such as FrameMaker to format such data (often aggregating it with other XML and non-XML sources) for publication.

Case Study: MFS Investment Management

MFS Investment Management (MFS) is a worldwide leader in providing investment services to individual and institutional customers, managing assets of \$122 billion. As the nation's first provider of mutual funds, MFS has a wealth of experience meeting clients' information access needs, and has a historical perspective on the changes of those needs over time.

The challenge of content aggregation

The information that MFS provides to their clients comes from a range of divisions and departments across the company. A single marketing brochure, for example, includes content aggregated from an array of sources: regulatory language from Compliance, marketing language from Corporate Communications, and valuation figures from analysts and staff across several other departments. This information also comes in multiple forms, including text, data, charts, and graphics.

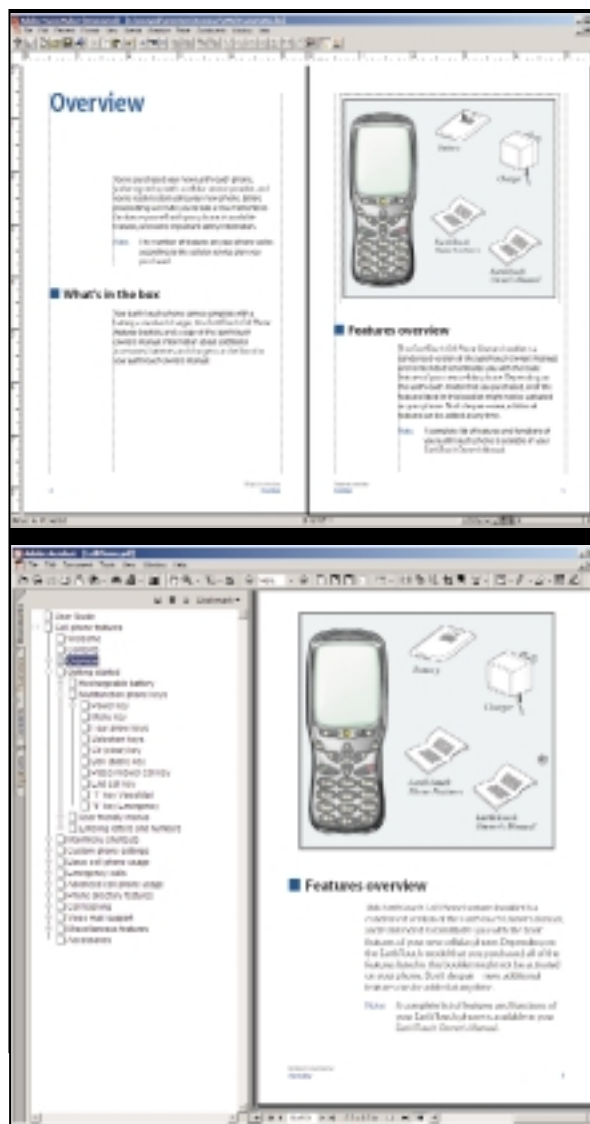


Figure 1 • FrameMaker lets authors see how their XML will look when published as print or Adobe PDF

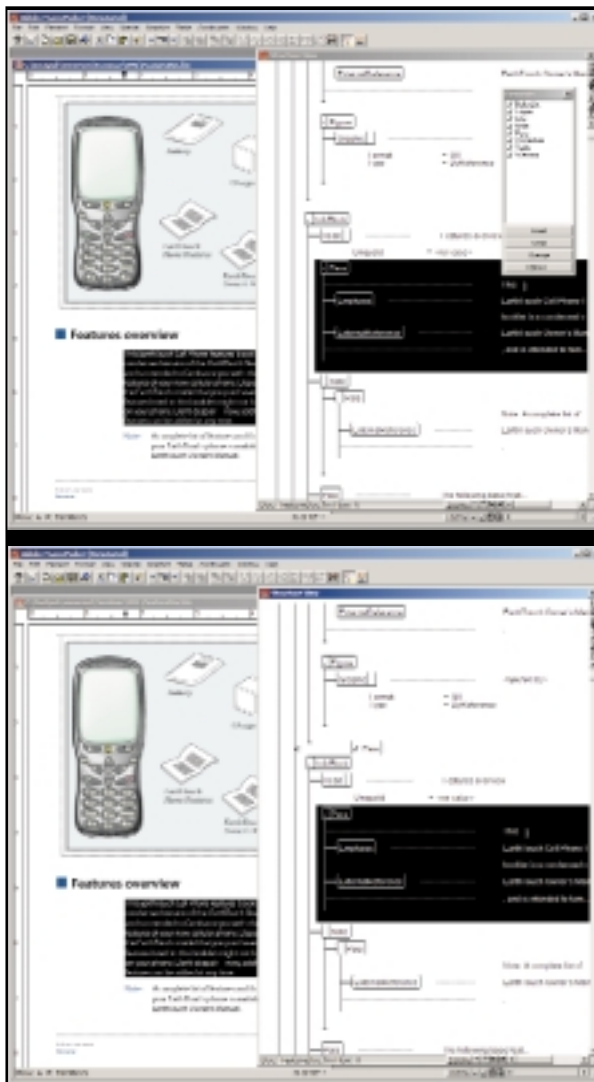


Figure 2 • XML authoring tools visually guide authors toward conformance with an XML DTD

The creation of such materials can be very time-intensive. The large number of document sources, the expanding range of required delivery formats, and stringent requirements for accuracy combine to make publishing particularly challenging for financial services organizations like MFS.

Creating performance-marketing materials for MFS was once a slow, cumbersome process: a number of dedicated employees using QuarkXPress and Xtags worked to assemble information and lay out pages. With such manual methods, marketing staff could produce only a limited number of publications.

A structured, automated solution

MFS dramatically improved this process by implementing an automated enterprise authoring and publishing system based around FrameMaker as a core authoring and publishing tool. The system uses PatternStream from Finite Matters Limited (FML) to connect FrameMaker to content sources such as relational databases (see Figure 3). The system helps MFS to more efficiently reuse content across departments, and has enabled the company to exponentially increase productivity: the new system, with fewer staff, publishes over 400 different marketing publications quarterly, delivered on paper and online in Adobe PDF format.

Going forward with XML

While the current system is automated and already provides many of the benefits of working with structured information, such as a template-driven publishing process, it is not currently based on XML. FrameMaker provides both structured (XML-based) and unstructured authoring and publishing capabilities, and initially the system uses the unstructured capabilities of FrameMaker.

Moving to an XML-based workflow by using the structured capabilities of FrameMaker 7.0 allows financial institutions to store all information in one place, from which it can be more easily reused. By moving to XML, companies will also be able to leverage additional tools for publishing XML, such as XSLT processors.

FrameMaker 7.0 uses the same document objects, such as character and paragraph styles, to format content in both unstructured and structured modes. This means that migration to XML will not require creating templates from scratch. Rather, a FrameMaker EDD (Element Definition Document) will be used to map XML structure to existing styles and layouts. FrameMaker can also help automate the conversion of unstructured content, such as word processing files, to XML.

Enjoying the benefits of an XML solution

An XML-based system can offer MFS several significant benefits. In the first place, a centralized content repository lets MFS store all information in one place, making it easily accessible across the organization, and making content aggregation a less arduous task. Using a central repository, and using XML-based techniques for managing document components, MFS can also improve the effectiveness of collaborative authoring, maximizing content reuse.

Benefits are apparent not only in the efficiency and effectiveness of internal processes, but also in the quality and throughput of published output. With a template-based publishing workflow built on highly structured content, MFS can count on consistent style and layout across documents of a given type. XML also expands the possibilities for multichannel publishing.

Case Study: Moody's Investors Service

Moody's Investors Service (Moody's) is a leading global credit rating, research, and risk analysis firm. They publish research covering more than 2,800 institutions in over 100 countries. Published content includes opinions, research, and ratings on fixed-income securities, issuers of securities, and other credit obligations.

Vast amounts of complex data

Moody's faces the challenge of analyzing vast amounts of information, and delivering the results of their analysis to clients in a variety of formats. For Moody's, information management is critical both internally and externally. Within the organization, analysts navigate through large amounts of information from a wide array of sources in the course of their research, while outside the organization, customers rely on Moody's for timely and accurate presentation of these analysts' assessments.

Until recently, Moody's created its documents using word processing applications and QuarkXPress. The data, including financial statistics, graphs, and tables, resided in databases throughout the company. Assembling lengthy reports took weeks and the efforts of many employees. "Traditional page layout tools weren't adequate to handle the volume and complexity of our data," says Ari Weinstein, assistant vice president and manager of the Publishing Solutions Group at Moody's.

Consolidating to a single source

Moody's overcame these challenges by migrating to a struc-

Mindreef

www.mindreef.com

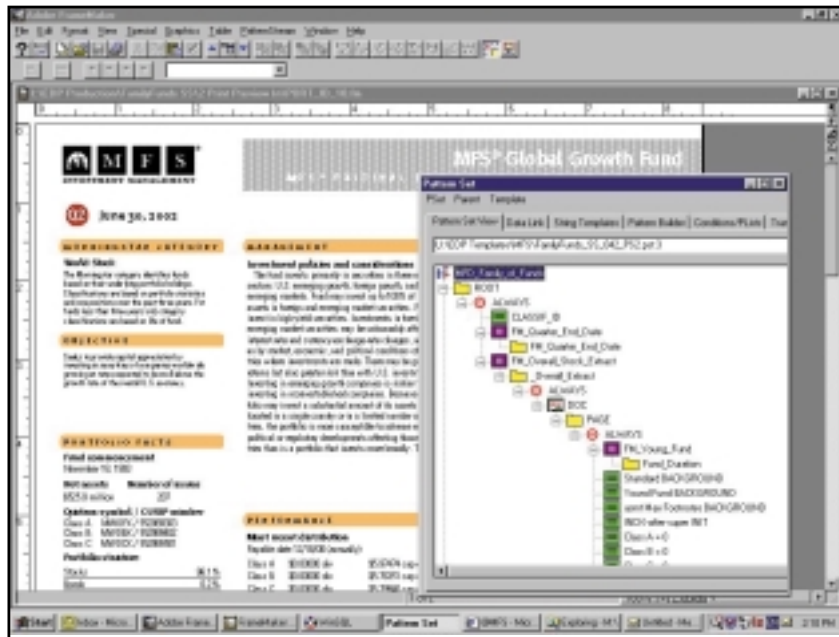


Figure 3 • FrameMaker and PatternStream

tured authoring solution based on a centralized repository. First, all data was consolidated into a single database and converted to SGML (Standard Generalized Markup Language). Moody's then made the move to an authoring and publishing tool that understands structured information: Adobe FrameMaker. FrameMaker provides both a powerful XML/SGML authoring environment that lets users navigate and edit structured information and a page composition engine that formats structured information for delivery to print or Adobe PDF formats.

Moody's is in the process of converting its SGML data to XML. The XML data will be pulled into FrameMaker templates for formatting, and then converted to Adobe PDF for online delivery. XML represents another step forward due to the increasing availability of XML tools that can be used in the content delivery process, as well as the increasing use of XML as a standard form of data exchange. XML will also potentially give customers the ability to perform more powerful queries for their own research. For

example, a customer could query all research publications and retrieve ratings for only the issuers in his or her portfolio. The fact that Moody's is using FrameMaker eases the transition from SGML to XML, as FrameMaker understands both, and (as with conversion of unstructured FrameMaker documents to XML documents authored in FrameMaker) uses common formatting methods for each.

Reaping the rewards of a structured approach

Now, when an analyst updates an opinion of an issuer in Moody's proprietary publishing system, the system automatically publishes the new information on the Web. At the same time, the system sends the information to the database as SGML. To revise the handbook, the production associate simply opens the FrameMaker file, which imports the up-to-date information.

After the FrameMaker document is complete, Moody's uses Adobe Acrobat to convert it to Adobe PDF for publication on the Web or submission to a print vendor. The Adobe solution boosted the productivity of Moody's production associates. "Producing a credit opinion handbook previously took four associates three weeks of work, and now with Adobe FrameMaker, it takes one associate just a day or two," Weinstein reports. Such time savings and

increased productivity result in cost savings as well.

Conclusion

Migration to structured authoring based on XML makes sense for industries such as financial services, where quick access to well-formatted information is an expectation of both internal and external customers. Payoffs such as reduced time to market, increased efficiency, and ultimate cost savings and competitive advantage can be well worth the effort of implementing an XML-based information management solution. ☛

AUTHOR BIO

Max Dunn is president of Silicon Publishing, a company located in Pleasanton, California, that provides electronic publishing services including the formatting of XML for print and online presentation. He has worked with a wide range of XML and SGML authoring and publishing tools.

MAXDUNN@SILICONPUBLISHING.COM

INDUSTRY COMMENTARY ~continued from page 5~

mation across components.

Our products also heavily leverage the many open standards that have been built on top of XML as well as XML's strength for describing both simple and very complex data structures. For example, all WebSphere Business Integration adapters, components whose job it is to connect to specific kinds of technologies and applications, use the open standard XML Schema, XSD, to describe the data format of the messages and events that they deliver and receive from the integration hubs they service. This is the case whether they are communicating with a process or a messaging hub. WebSphere Business Integration leverages the same XML standard for metadata description between our tooling and runtime components. Our native support for XML also makes our ability to support the many growing industry standards that much easier.

Moving Forward

Are there application areas or industries in which XML hasn't realized its potential? Without a doubt, although I don't know of a single industry that has not already or does not have plans to leverage XML for business process optimization. Some industries are further along than others, obviously, but all are moving in a direction to increase reliance on XML for sharing data with partners and customers. Nonetheless, XML still is characterized by having greater performance and processing overhead when compared to compiled application technology. IBM will continue to invest research funds to optimize its XML technology. An obvious example is developing ever more efficient XML parsers in order to minimize that overhead even further.

What are the most strategically important XML-based initiatives for business

integration moving forward? I'd have to say Web services. The potential of Web services closely parallels that of XML – both are truly platform independent and both have or are gaining broad acceptance by the leading vendors. Thus Web services and XML have the same potential to be universally accepted – pervasive standards leveraged in all areas of the IT industry. It's not surprising that XML already plays a central role in Web services standards. ☛

AUTHOR BIO

Rachel Helm serves as director of product market management, WebSphere Business Integration, at IBM, where she has overall responsibility for product planning for IBM's WebSphere business integration middleware. Rachel has been involved in the software industry for more than 20 years, working in product planning and management, product development, and consulting.

RACHELMH@US.IBM.COM

JavaOne

java.sun.com/javaone/sf



Enabling Integration of Internal Data and External Business Information

New technologies deliver accurate snapshots

Few executives will deny that critical business decisions must be based upon timely, accurate information. Historically, however, delivering actionable business intelligence to the appropriate person at the right moment has been difficult, largely due to the fact that internal corporate information commonly resides in isolated islands of information, with each functional area maintaining its own repository of data, with its own data taxonomy.

To address this problem, companies are increasingly investing in customer-focused enterprise applications to link together internal databases and deliver a snapshot of interactions between the organization and its customers and prospects. While enterprise systems such as Customer Relationship Management (CRM) and Enterprise Information Portals (EIP) can offer a comprehensive internal view, they typically deliver few details about a customer's changing business fortunes, strategic decisions, or industry trends. This has the potential to limit individual use and minimize the ROI.

The most current business information typically resides in external databases — such as SEC filings of financial data, press releases, news coverage, trade press articles, industry reviews, and analyst reports. To create a complete profile of customers and prospects, therefore, companies must be able to link internal data with high-quality external business information and deliver the two seamlessly to decision-makers on a daily basis.

By effectively merging internal and external data, companies can:

- Target the most qualified prospects
- Focus on building stronger relationships with the highest-value opportunities
- Minimize business risk

- Cross-sell more effectively
- Extend relationships across multiple buying groups more easily
- Maximize the return on CRM systems, information portals, intranets, and other enterprise applications

Enabling Technologies Simplify Data Integration

Corporations have long recognized the value of combining comprehensive internal data with external intelligence. The complexities of integrating disparate sources, however, have traditionally limited practical implementation.

External information can be manually added to an internal database. This approach, however, is time consuming and offers only a short-term solution unless the information is continually refreshed. The value of the combined internal/external snapshot immediately begins to fade as data ages and new events occur outside the corporation.

Another alternative is to build custom interfaces between various databases and business applications. The challenge is that integrating data from applications written in different languages and operating systems requires a significant investment of time and money, as well as a very broad range of specialized programming skills. And once the initial integration is complete, significant work is required to maintain and extend the interface over different software releases.

Fortunately there is now a more efficient and cost-effective approach for linking internal data with external business information. In the past, the process of integrating disparate applications has been limited by steep technology barriers and software compatibility problems. However, XML, SOAP, and Web services have emerged as viable technologies to

perform this type of integration.

Web services allow any computer to reach out to other computers for the information and applications it needs to perform a task. (Figures 1 and 2 show the difference between proprietary systems and Web services.) XML provides an industry standard for transferring information between computers and applications. SOAP is an industry-standard wrapper that allows computers to exchange XML messages effectively.

Web services, XML, and SOAP play a key role in data integration because they deliver significant advantages, including:

- Making integration easier
- Allowing for seamless integration
- Achieving consistent data presentation
- Building information-driven triggers

Making integration easier

The importance of XML, SOAP, and Web services extends far beyond the Web and the display of information within a browser. These standards make it easy to exchange information between disparate computers, databases, and applications — regardless of the application, programming language, or operating system.

As long as two applications agree on a consistent set of XML tags, they can easily exchange information. With the proliferation of standards based on XML this may initially seem like a major hurdle. Thousands of XML-defined standards have been proposed, most of which have little hope of achieving critical mass. However, those that are widely supported by end-user communities, industry associations, vendors, and standards bodies (such as XBRL for financial reporting) are clearly useful and worth considering for specific internal-external data integration tasks.

Since an application can be set up to request tagged data from outside a corpo-

AUTHOR BIO

Patti Purcell, senior vice president, products and solutions, leads OneSource's product innovation and embedded solutions initiatives. She has more than 20 years of experience with enterprise software, professional services, and information products at leading technology companies. Most recently, Patti was vice president, professional services, for Interliant, a publicly traded provider of managed infrastructure solutions.

rate firewall, internal business applications can pull in updated external information and immediately display it within the application. With XML and Web services, integrating multiple applications that formerly required significant time and professional service dollars can be accomplished much more simply. And the XML code is easily portable to updated versions of the same application.

Allowing for seamless integration

XML data is format independent, so external information can be merged with internal data and displayed in any format consistent with a defined set of corporate standards. This seamless integration makes it easy to present external data that is most relevant to an individual functional group or business operation, in context with internal data.

Users can rely on their everyday applications and receive a much richer display of relevant business information, including news stories, current financial data, industry trends, updated executive biographies, and detailed corporate family information.

Best of all, integration can occur behind the scenes, with automatic updating of account profiles, financial models, credit scoring applications, etc. Not only are extra steps streamlined from key business processes, but the traditional risk of errors caused by manual data entry is virtually eliminated.

In addition, automatic updating can occur with unstructured information – such as news stories and analyst reports – to track and keep you abreast of changing details about prospects, customers, and competitors.

Achieving consistent data presentation

Incorporating external data within existing applications ensures that data is displayed in a consistent and familiar format, thereby minimizing unproductive research, improving productivity, and allowing end users to rely on a single application with consistent field placement for critical pieces of data. For example, sales managers can routinely refer to a single field within their CRM system to review the latest revenue data.

The importance of consistent display extends beyond just applications to custom reports, presentations, and individual forms. Automatic data integration makes it easy to update regular reports, sales presentations, company profiles, risk analysis worksheets, financial comparisons, and any other material.

Building information-driven triggers

Integrated data offers far more than just

formatting advantages. Once external information is incorporated within enterprise applications, it can be used to trigger alerts that drive specific business actions. An alert can be as simple as notification of a recent corporate announcement, or as complex as an elaborate financial simulation that immediately integrates new financial results. Alerts ensure that users receive the information they need to do their jobs effectively.

Given that XML is device independent, information and alerts can appear on any platform, including desktop PCs, wireless phones, and PDAs. With XML-tagged data, business users can receive the information they need in the format that best meets their needs at any given point in time. For example, salespeople can receive customer updates within a CRM application on their office PC, as an alert on their cell phone, or as an e-mail on their PDA.

The Business Information Taxonomy

While XML, SOAP, and Web services play an important role in easy and seamless integration of internal and external data, the task of creating a truly successful, integrated information repository requires a level of expertise and supporting tools not often found inside the corporation. This reality is driving many companies to seek comprehensive business information solutions that provide all of the tools and services to integrate internal and external data within CRM systems, portals, and intranets.

In choosing a business information solution, companies must carefully evaluate a number of factors, including the range and quality of information provided, the experience and expertise of the information vendor, and, perhaps most important, the vendor's business information taxonomy. While XML and Web services help deliver data, taxonomies provide the framework for classifying, organizing, and integrating a wide range of critical business information.

The challenge is that taxonomy solutions are generally limited to a single approach: highly structured data (ZIP codes, SIC code, etc.) or unstructured text documents (news articles, press announcements, analyst reports, etc.). The unstructured taxonomies run deep on business and topic terms but don't perform well for tagging companies – the key entity to a customer-focused application. The structured taxonomies are typically too narrow in focus and don't cover the universe of information required for customer-facing processes. When forced to choose one taxonomy approach versus the other, businesses come no closer to integrating isolated islands of

data and arming sales and marketing professionals with comprehensive intelligence.

When outsourcing data integration projects, companies should rely on vendors who offer a classification system that efficiently organizes both structured and unstructured content.

Conclusion

Companies need to employ the best market intelligence if they are going to succeed in today's increasingly competitive environment. By using XML, SOAP, Web services, and a trusted provider of business information, a company can seamlessly integrate internal and external

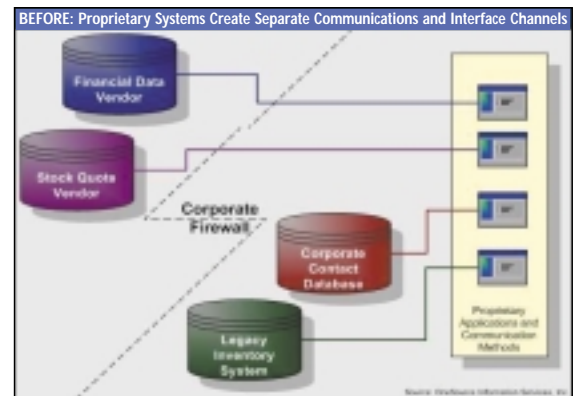


Figure 1 • Proprietary systems

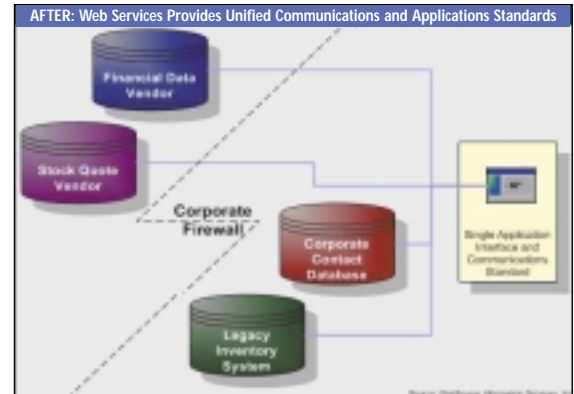


Figure 2 • Web services

data to provide complete snapshots of their customers, prospects, and competitors. These can be continuously refreshed with current information, so they never become irrelevant.

Accurate snapshots allow companies to make more informed decisions. They help marketing identify new opportunities and precisely target high-value prospects. And they help sales respond faster, prepare for customer visits more efficiently, dramatically improve close ratios, and generate more revenue per employee. ☛

■ PATTI-PURCELL@ONESOURCE.COM

Applying Design Patterns to Web Services Architectures

A few well-known patterns can simplify your development efforts

WRITTEN BY
CHRIS PELTZ

From the beginnings of the well-known “Gang of Four” design patterns book to more recent publications on J2EE design patterns, the software industry has always tried to find ways to design frameworks, ideas, and concepts that could be used repeatedly. With the introduction of Web services technologies, the need for design patterns remains the same.

The good news is that software architects today can apply many of the existing design patterns to Web services. Use of these patterns can greatly help the architect in building scalable, reliable, and robust Web services architectures.

This article takes a look at four of the more well-known design patterns and discusses how they can be applied to Web services architectures. The following design patterns are examined here:

- **Adapter:** The Adapter pattern is often used to provide a different interface to an existing software component to make it more compatible to a client. This pattern can be used to wrap existing technologies, such as an HTML interface, to make it more compatible with SOAP.
- **Facade:** The Facade pattern has been used frequently in J2EE to reduce the coupling between client and server. This same design approach can be used to create the right level of granularity in the Web services that are exposed to a client.
- **Proxy:** The Proxy pattern is used to provide a placeholder for another object. Here, I show how the proxy pattern can be applied to a number of areas in Web services design, including client proxy generation, mobile processing, and Web services testing.
- **Controller:** The Controller pattern has been used as a way to separate presentation from data. It is often used in a Model-View-Controller (MVC) architecture. This article shows how an architect can introduce Web services to an existing MVC-based design.

While this is only a sampling of the many design patterns available today, it will hopefully give you a better sense of the

value patterns can bring to Web services design and development. So, let's begin by taking a look at how the Adapter pattern can be used to incorporate incompatible technologies into a Web services architecture...

Managing Incompatible Interfaces

An organization will often look to existing software assets to exploit Web services technologies. In some cases, these existing applications utilize languages, protocols, or platforms that are not compatible with the target Web services platform. Managing these incompatible interfaces can be solved through the use of the Adapter pattern. The intent of this pattern is to “convert the interface of a class into another interface clients expect” (*Design Patterns: Elements of Reusable Object-Oriented Software*). It would typically be used to provide compatibility between multiple platforms or systems.

In the design of Web services architectures, the Adapter pattern can be used to expose existing components as Web services. It's important to remember that any given Web services platform will only support a set number of languages and protocols. This pattern can be used to wrap any incompatible interfaces with one the Web services platform understands. For example, Figure 1 illustrates how a C++ interface and the HTTP protocol can be utilized on a J2EE-based platform.

Two approaches can be taken to integrate a C++ component. If you have a large amount of code written in this language, you might want to invest in a C++-based Web services platform, such as the Systinet WASP Server for C++ (www.systinet.com). But, if you have standardized on a J2EE-based platform, you can typically only expose Java-based components, e.g., Enterprise JavaBeans, directly as Web services. One solution to this problem is to write a Java wrapper class that can interface with the C++ component over JNI or sockets.

Taking a look at the HTTP example, there might be a case in which a company has already exposed a software asset with HTML over HTTP. These Web-based applications might make excellent candidates for Web services. Web services and Web applications already share a common protocol – HTTP. The main difference is the use of HTML instead of XML. In this sce-

nario, an Adapter could be written that invokes the appropriate URL Web pages, and repurposes the HTML content as XML. Listing 1 shows a simple example of how this could be done.

While this approach may work in some cases, you also have to remember that every HTTP-based application may not necessarily make a useful Web service. This is a similar design problem that existed with mobile application design. In the beginning, many developers were “transcoding” their existing Web site to quickly get their Web application available through a mobile device. But, many of these applications were not directly suitable for the mobile user.

Regardless of the design choices you make, the important thing to remember is that you don’t have to start from scratch with Web services. You can leverage many of your existing software assets, and the Adapter pattern is one approach to help resolve integration issues. At the same time, you have to be very careful about trying to provide the quickest Web services solution. This often will not always meet the needs of the clients who have to use the service.

Creating Business-Level Web Services

To meet the needs of the business, you have to design the right level of granularity in your Web services. Choice of granularity can impact the usability, performance, and management of the services. A fine-grained approach typically maps each service to a low-level component. A coarse-grained approach exposes services that map to a particular business need.

We can illustrate the differences between these two approaches by looking at XMethods (www.xmethods.com), a popular Web services Web site. On this site, you can find services that deliver news, stock quotes, and text translation services. These services provide a specific function to the caller, but don’t necessarily fulfill a business need. By combining these services (e.g., connecting the news feed service to the text translation service), we can begin to deliver more coarse-grained services that deliver additional business value.

The Façade pattern can often be applied to build coarse-grained services. This pattern has been most recently applied with J2EE through the Session Façade pattern (*Core J2EE Patterns: Best Practices and Design Strategies*). In J2EE, there are entity beans that typically model data and session beans that model business logic. It was common for developers to have clients talk directly to the entity beans, resulting in a number of fine-grained invocations to retrieve data. A façade could be written around the entity beans by creating more coarse-grained components represented by session beans.

This same design approach can be leveraged for Web services. Rather than providing a direct one-to-one mapping to every existing software component, you can encapsulate multiple components into higher-level, coarse-grained services. Figure 2 shows how the Façade pattern can be used in this regard. The individual components can be any software technology, including Java, EJBs, JMS, C++, and CORBA. They can even be other Web services that were already exposed with a WSDL interface. This is where Web services orchestration standards (BPEL4WS, WSCI, etc.) become important.

A number of key benefits are gained by using the Façade pattern with Web services:

- **Consistent interfaces** You are building consistent interfaces to your low-level application components, interfaces that more closely match the specific business requirements. As a result, you can shield the clients from having to know some of the back-end implementation details.
- **Centralized management** Use of a façade can also help facilitate, control, and manage your Web services environ-

ment. The façades will act as centralized access points, handling issues such as security, transactions, and management. Without these “umbrella” services, the individual components, or the client itself, would have to handle this.

- **Improved performance** This is one of the biggest benefits of this pattern. By reducing the number of client access points, you reduce the number of network calls required by the client. Data is bundled together, typically mapping to a specific business service needed by the client.

As was mentioned in the previous section, it’s very important to remember that not every existing component will make a good Web service for the consumer. In addition, use of object-based technologies such as EJBs and CORBA still serves a valuable purpose in an infrastructure. The value will come when you can identify how to best combine and aggregate these components to deliver real value to your customers, partners, and suppliers.

Isolating Complex Processing

The Proxy pattern is another well-known pattern that can be used to isolate or hide complex processing. The intent of the pattern is to “provide a surrogate or placeholder for another object” (*Design Patterns: Elements of Reusable Object-Oriented Software*). It can be applied to Web services to simplify the interaction between Web services components.

The most common application of this pattern is to hide the complexity required to construct SOAP messages. Almost every Web services platform available today uses some form of this pattern to shield the developer from this complexity. For example, Apache Axis (ws.apache.org) provides a WSDL2Java tool that will automatically generate client proxy code containing the SOAP processing logic. The following code snippet illustrates how a client can communicate with a service using an Axis-based client proxy.

```
public static void main(String[] args) {
    WeatherService service = new WeatherServiceLocator();
    Weather weather = service.getweather();
    Forecast forecast = weather.getWeather(args[0]);
}
```

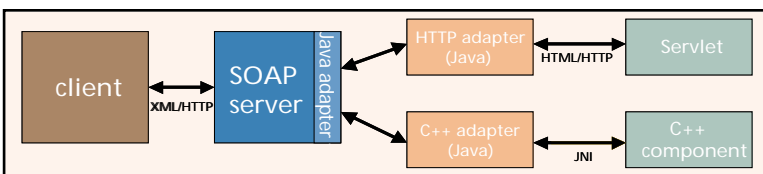


Figure 1 • Applying the Adapter pattern

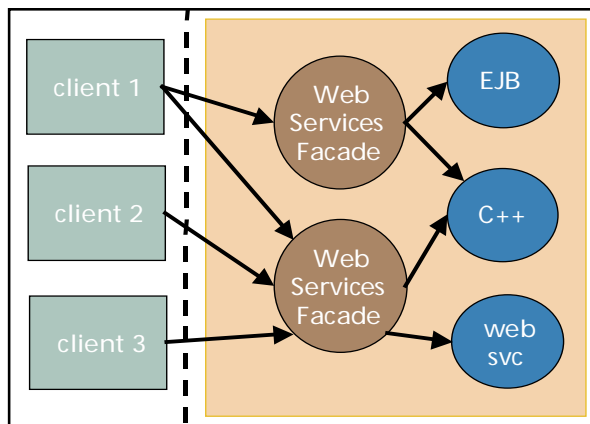


Figure 2 • The Façade pattern


```
System.out.println("Forecast is " + forecast.getForecast());
}
```

The Proxy pattern can also be used to move complex processing from the client to a back-end service. This is often useful when dealing with lightweight clients, such as mobile devices. Instead of requiring the client to perform the processing, this work can be moved to a proxy server. The proxy server acts as an intermediary between the client and the Web service. The client can have a very thin communication layer, with the actual SOAP invocation being done by the proxy server. This approach can help reduce the number of network calls required by the client to communicate with the service.

Another function for a Proxy is to serve as a standardized interface for a collection of back-end services. For any given deployment, there might be a handful of services available on a server. Rather than giving each service its own entry point, a proxy server can be developed that consolidates these messages into a standardized interface. Again, the benefit is simplification of the communication model required on the client side.

This notion of standardized interfaces can be applied in the Web services testing domain. During the testing phase, your components may need to communicate with third-party services that are outside your control. A common scenario is an e-commerce-based service that includes a payment-processing step. As part of this process, the consumer's credit card may be charged for a given purchase. However, during the testing of the service, the developer wouldn't want to interface with the real payment system. Instead, they would typically write a "dummy" service that would be functionally equivalent to the original.

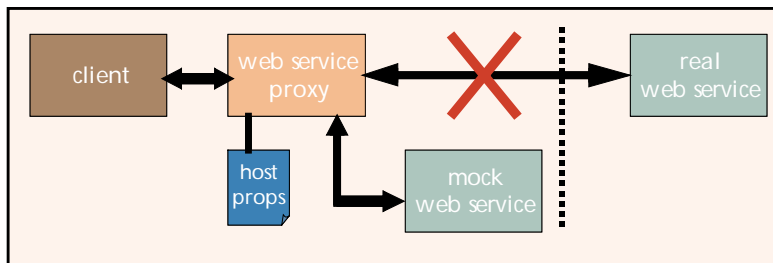


Figure 3 • Using mock objects to test Web services

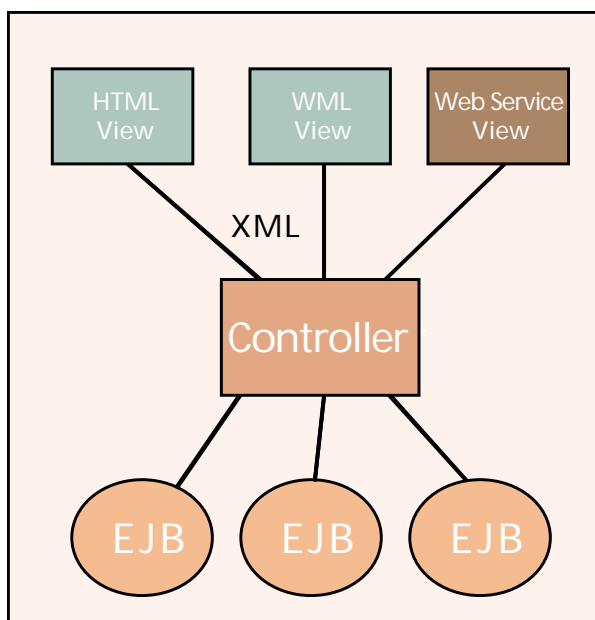


Figure 4 • Leveraging the MVC architecture

To manage these external dependencies, a developer can use a technique called mock objects, an Extreme Programming (XP) approach to unit testing ("Endo-Testing: Unit Testing with Mock Objects"). The basic idea is that you create a mock object that takes the place of the actual object. The mock object will look and act just like the original component. In our example, a mock object could fulfill the needs of a credit card service without requiring the developer to invoke the real system during testing.

Figure 3 illustrates how mock objects could be used with Web services testing. In this approach, the Proxy pattern is used to encapsulate the SOAP messaging logic. The Proxy is further enhanced through the use of an XML configuration file. The purpose of this file is to indicate the host and port number where the Web service is located. If the original Web service and the mock Web service conform to the same Web service interface (WSDL), the only change that has to be made is in this configuration file. A developer could quickly update the configuration file to indicate which version of the service to use.

Leveraging Existing Frameworks

The previous approach of using the Proxy pattern to encapsulate complex processing logic provides another important benefit – leverage. In fact, patterns exist to provide reusable concepts or ideas that can be applied to solve new programming problems. As a Web services architect, you need the ability to quickly recognize where and how you can leverage your existing components, systems, and infrastructures. One important area to focus on is the use of existing frameworks that can be applied to Web services architectures.

One of the better-known architectural frameworks is the Model-View-Controller (MVC) architecture. Originally a design applied to the Smalltalk language, it has been frequently used in the design of Web-based architectures. The general design approach around MVC is to clearly separate the key components of the architecture. These components are:

- **Model:** Contains the information, or data, required by the application
- **View:** Manages the presentation for the application
- **Controller:** Acts as an intermediary between the client and the data

If architected correctly, an MVC approach will allow an organization to expose multiple interfaces to the same set of data. One application of this has been in J2EE architectures, where Enterprise JavaBeans (EJBs) are the model, JSP and HTML pages are the view, and servlets are used to manage the navigation between Web pages. With this separation, a developer can easily plug in new presentation modules that support different device types (e.g., HTML, WML, etc.).

So, how can Web services leverage the MVC architecture? If you think about Web services as just another view on the model, a Web services "presentation" can easily be plugged into an existing MVC architecture. Rather than sending HTML or WML back to the client application, the Web services view would construct XML or SOAP messages through interactions with the Controller (see Figure 4).

This solution can be somewhat hard to understand because a Web service doesn't really have a presentation – it's really more of a model in MVC. But, if you think about it, Web services can be looked at as another view (or interface) on an existing information model. In this case, the view is represented as SOAP. Obviously, if you have an existing architecture that generates XML from the controller, it is even easier to extend your application to support Web services. The architecture will still have to design a presentation layer that can present the SOAP-based information

to the consumer (e.g., through a Web-based portal).

The overall intent here is to illustrate how you can easily leverage your existing architecture to support Web services. While we have just talked about MVC in this example, similar approaches can be applied to other software frameworks you might be using. This approach might make sense if you have an existing set of components you want to expose as Web services, but it might not be appropriate for new Web services projects being designed from the ground up.

Conclusion

If you are beginning to explore the use of Web services, it's important to remember that it is not a revolutionary new approach. Web services build on many of the existing distributed technologies and paradigms. Using Web services does not necessarily mean that you have to throw away all of your existing components and start from scratch. In fact, you can discover opportunities to leverage many of your existing software assets and expose them with Web services interfaces.

The right amount of design and analysis should be performed to determine what would make a good Web service. Not every existing object or API can map directly to a Web service, and it's important not to fall into the trap of delivering the quickest solution that might be offered by the Web services platform. Choosing the right level of granularity can help deliver usable, well-defined, and scalable Web services to your customers, suppliers, and partners. These services can be combined and aggregated to provide even more business value.


Value can also be delivered by leveraging many of the existing design patterns that have been developed for object-oriented systems. This article showed you how a few well-known patterns can be applied to Web services architectures. With these specific patterns, a Web services design can simplify the

LISTING 1 • Adapting to an HTML interface

```
// Setup URL string to contact service provider
String url = http://localhost/sendsms;
String urlString = "http://localhost/sendsms?"
    + "msisdn=" + msisdn + "&text=" + txt;
URL url = new URL (urlString);

// Setup BufferedReader to get HTML content
InputStream content = (InputStream)url.
    getContent();
BufferedReader in = new BufferedReader (new
    InputStreamReader (content));
String line;
while ((line = in.readLine()) != null) {
    // do appropriate re-purposing of HTML
    content...
    pw.println (line);
}
```

Download the Code
www.sys-con.com/xml

development effort required to take a set of existing software assets and expose them as business-level Web services. 

References

- Gamma, E.; Helm, R.; et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.
- Alur, D.; Crupi, J.; et al. *Core J2EE Patterns: Best Practices and Design Strategies*. Sun Microsystems Press, 2001.
- Mackinnon, T.; Freeman, S.; et al. "Endo-Testing: Unit Testing with Mock Objects." www.mockobjects.com, 2000.

AUTHOR BIO

Chris Peltz is a senior software consultant in HP's Developer Resources Organization (devsource.hp.com), providing technical consulting on J2EE and Web services architectures. He brings over ten years of software development experience in helping customers select technologies, tools, and platforms for building enterprise applications. Chris has presented on Web services at HP World, Web Services Edge, and JavaOne.

Ektron
www.ektron.com/xmlj



HOME



Enterprise Solutions



Content Management



Data Management



XML Labs

INTRODUCING Microsoft InfoPath 2003

Part 1

WRITTEN BY
THOM ROBBINS

Reduce your 'mountain of paper'

A few weeks ago I was meeting with the CIO of a local health care customer and his IT staff. They were explaining the various technology initiatives and projects that were occurring over the next year. What the CIO was the most proud of was that he had declared this a year of integration projects.

He had followed the IT trends closely and was seeing that now was the time to ride the Web services wave. With the release of Visual Studio .NET and Windows Server 2003 he felt comfortable in the platform. He and his staff were focused almost exclusively on the development and deployment of a Web services architecture to tie their back-end hospital systems and other various customer-facing systems together. The main goal he had dangled to his board was that he could reduce cost and increase efficiency through process automation and back-end integration. He had explained to them that Web services provide a clean and easy way to tie together their various back-end systems regardless of platforms.

As the CIO described it, their biggest remaining problem was the "mountain of paper" that had become the mainstay of their organization's daily life. Each piece of paper represented some portion of a vital health care process required for the hospital to function properly. They ranged from patient admission to nursing orders. It became clear through our conversation that ASP.NET combined with VB front-end applications wasn't the sole clear answer for front-end data collection. Web services clearly answered the back-end integration requirements, but there was still a definite need for offline access and a quick and easy way for data collection. I explained that a new product called InfoPath 2003, which is part of the new Microsoft Office System 2003, was designed to solve that very problem. In the first part of this two-part series I'll show how you can use InfoPath and Web services to solve not only this CIO's problem but similar problems within your organization. In Part 2 I'll explore further ways that you can integrate InfoPath with Microsoft BizTalk Server 2002 to create robust workflow applications.

The design goal of InfoPath 2003 is to streamline the process of gathering information by enabling teams and organizations to easily create and work with rich dynamic forms. The real key to InfoPath is that the data collected can

easily be shared and integrated with a variety of back-end business processes because of the native support of customer-defined XML. The use of XML as the basis of InfoPath provides a built-in reusability for XML-enabled applications. Additionally, InfoPath provides the native ability to consume XML Schemas, Web services, SQL, and Microsoft Access databases.

What Is an InfoPath Solution?

By definition an InfoPath solution consists of either a form template or a set of files that are combined to provide the necessary semantic information to the InfoPath clients to generate a form and output the resulting XML (see Table 1). These files can be packaged separately or combined into a single .xsn file. These groups of files are in a hub-and-spoke relationship with the form definition file providing the entry point. This file, by default named manifest.xsf, consists of an XML document that uses the namespace and associated schema of <http://schemas.microsoft.com/office/infopath/2003/solutionDefinition>.

An InfoPath solution can be created using the design mode of the InfoPath 2003 client or authored independently. The client application supports round-tripping so that the form definition file and the various XSLTs can be modified by the solution designer and then reloaded within the InfoPath client. This gives additional design and extensibility capabilities outside of the InfoPath environment.

An InfoPath solution is run by loading an XML instance that is associated with a given form. The information in the form definition allows InfoPath to display the XML data and to define the user interface, types of interactivity, and validation rules. During processing, the XML data instance has a processing instruction (PI) pointing to the form definition file that leads to the loaded form along with any specific instance data. The form definition file is really the key to the InfoPath document structure and contains a variety of information that is required for the form to load and process. This includes a unique forms identifier and metadata that is used for publishing and deployment information. The form definition file is also where the XML DOM schema, business logic and event handlers, view definition, and user interface are stored.

When you create a form in design mode using the InfoPath client you are actually creating a form template. Every

InfoPath form is based on a form template. Form templates or the .xsn files define the layout and functionality of a form including the XML Schema that determines the structure of the data when the form is filled out by a user. Whenever a user fills out a form, the form references the form template it is based on regardless of whether the form is local, on a network share, or installed in a Windows Sharepoint Services forms library.

The InfoPath security model is similar to that of Internet Explorer. Form templates by default run in a sandboxed environment and this protects the system from potential malicious code. As an alternative, InfoPath forms can be deployed through an installation program (MSI); this will grant them a higher permissions level as they are running on the local machine. If you want to identify the source of the form you can look at the bottom of the InfoPath client. This will show the running location of the InfoPath form. A URL identifies the form as hosted on a Web server or by a URN if they are deployed locally.

Creating a Solution

Now that we understand the basics of an InfoPath solution, let's go ahead and create a form. When trying to decide what form to implement using InfoPath, my health care customer decided to design a form that had a corresponding Web service that captured new patient information already developed and implemented in their production environment. This Web service acted as the front-end data collection point for their back-end MUMPS system. So they wanted to extend the Web service into an InfoPath solution that could be filled out by either patients from a kiosk or hospital staff that collected the required information during the interview process.

InfoPath provides several ways to develop forms. In this example, we already had a Web service that would act as the data source. In order to create the connection between InfoPath and the Web service you simply select the data source wizard from the design menu and select Web service as the provider (see Figure 1). The wizard accesses the WSDL of the selected Web service and generates a list of fields based on the service description of the Web services. In turn this becomes the generated list of fields that we use to create our InfoPath form. If InfoPath is unable to access the complete data structure returned by the operations of the Web services form, designers are prompted to specify some sample values for all the input arguments needed by the operation. This data is then used to make a call into the selected operation and then design the data structures based on the received output.

InfoPath is capable of building a form template from pretty much any arbitrary XML Schema that is valid according to

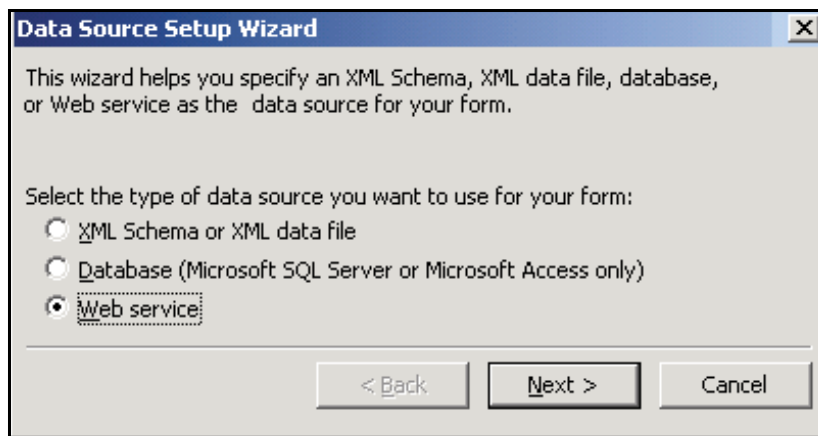


Figure 1 • Data source setup wizard

the standard W3C specification. There are some schema constructs that InfoPath is not capable of supporting in this version. You will want to keep these in mind as you develop your Web services:

- Schemas that define abstract complex types
- Schemas that define content models with any nodes that have a maxOccurs greater than 0
- Schemas in which a choice node has a direct child sequence or list

Within the InfoPath design environment all work is done through the various task panes. Each of the task panes represents a specific design activity. These include layout, controls, data source, views, and publishing. After you have completed the data source wizard, InfoPath will place you in the data source task pane with a list of the design fields and a blank form to begin placing your data elements.

Build the Form

Now that we've successfully created a connection to our Web service and created an instance of our XML data we are ready to build our data entry form. The easiest way to build a form is to build around table structures, similar to building a Web page. Tables will help to control the formatting and provide an easy alignment for the various fields that you can place on your forms. The layout task pane provides a variety of table options that you can drag onto your blank form.

InfoPath provides two important types of sections or tables that can be used to improve the formatting of your form. The first is the repeating table. In this table each column represents a field and each row represents an additional occurrence of

File Type	Extension	Description
Template definition (manifest)	.xsf	An InfoPath generated XML file that contains information about other files
Schema	.xsd	The XML Schema files that are used to constrain and validate the XML document files
View	.xsl	Presentation logic files that are used to present, view, transform data
XML Sample File	.xml	XML file that contains the default display data
XML Component Template	.xct	XML file representation of the editing controls for design mode
Presentation	.htm, .gif, .xml	Files used in conjunction with view files to display the custom interface
Business Logic	.js, .vbs	Script files used for event handlers and data validation
Binary	.dll, .exe	COM components that provide additional business logic
Packaging	.xsn	A compressed file format that packages the form files

Table 1 • An InfoPath application creates a specialized class of XML documents, possibly including many different types of documents.

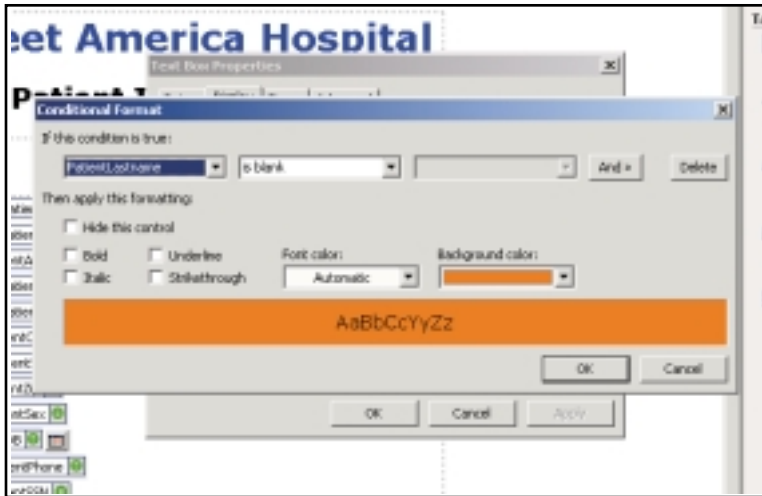


Figure 2 • Conditional format

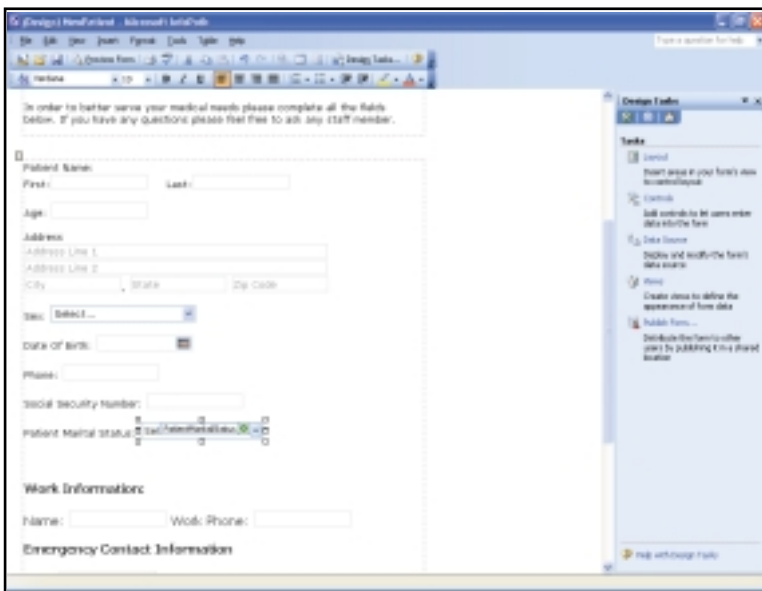


Figure 3 • Form in design mode

the group of fields. This would be an easy solution if, for example, you were looking to collect a series of patient insurance information and didn't want to limit the number of items that someone could enter. The second is the optional section. These are sections that can be inserted and removed by users while they fill out forms. As a form designer you can specify whether fields or groups of fields should appear by default in the blank form or if they should appear only when the user inserts them. For example, if I wanted to allow a notes section on my form I could allow the user to determine if he or she wanted to include the section or not. This allows the user to remove the entire section without having to delete the text and leave a blank field.

For the patient information form there were two sections that we needed to develop. The first was a title or header section that contained the form name and instructions for completing the form. The second area was the actual data area. This area contains the fill-in fields that users will complete as part of the form.

Moving the data fields on the tables is an easy drag-and-drop operation from the data source task pane to the columns within the table. It is important to understand that based on

the data types specified by the WSDL of the Web service InfoPath will associate a specific control type. For example, if you look at the date of birth field you will see that a date picker control was automatically selected. If you decide that you want to change the default associated control you can right-click and select the specific control type that you want. Additionally, if you want to validate which field a particular fill-in field is bound to, you can hover your mouse over the field. A pop-up window with a green indicator and a schema field name will identify that your field is properly bound to the data source.

After placing all the required fields and saving the form template you are ready to fill out and submit the form to the bound Web service. In addition to Web service submission end users also have the ability to export their form to a static Web page, to Excel, or to someone through e-mail. As you begin developing InfoPath solutions, e-mail provides an easy out-of-process workflow step—for example, if you wanted to notify someone via e-mail that a particular patient has entered the physician's office.

Data Formatting and Validation

At this point we've created a fill-in form that could be distributed and used. The problem is that we haven't included any of the formatting or business logic needed to validate the entered data. InfoPath provides designers with the ability to control the formatting of plain- and rich-text boxes, sections, and repeating tables based on a set of user-defined conditions. As you create a condition, you specify the font, styles, background color, and even whether to hide the specific control. Conditional formatting is stored as part of the XSLT and is applied to form fields whenever the specific set of conditions is met. For example, one of the requirements for the patient information form was that required fields were to have a red background. Conditional formatting statements are entered through the field's display options, by selecting the conditional formatting tab. Within the tab, building the expression is done by selecting the various fields and filling in the required conditions (see Figure 2).

One of the main requirements of most data-driven forms, in addition to conditional formatting, is validation. InfoPath provides three levels of validation: schema-based validation, declarative or rules-based validation, and custom script. Depending on the complexity of your form it's possible to have all three levels implemented on a particular field. Schema-based validation and then declarative script-based validation are defined based on an InfoPath event model, so the forms developer has full control over when this occurs. InfoPath allows a form to be saved without successfully passing validation but will prevent submission to the data source until all checks are successfully completed.

For our patient information form we used an existing Web service. This allowed us to take advantage of schema-based validation. Schema-based validation occurs whenever a user fills in a form. After data is entered in a field and the user moves to another field the data is immediately checked against the schema. If the data fails the schema, check the marked field with a red dashed border. The user can then right-click the marked field and be given feedback as to the data type or schema requirements. As we are bound to a Web service our schema is predefined. For example, if I entered text into the age field, which is defined as an integer, a schema validation error would occur. By default InfoPath will check for an invalid data type, specified range, and required fields as defined within the schema definition.

Event	Description
OnVersionUpgrade	Occurs when the document is opened and when a newer version of the document has been detected
OnLoad	Occurs when the document is first open
OnSwitchEvent	Occurs when the user changes views

Table 2 • Events associated with document lifestyle

Event	Description
OnVersionUpgrade	Occurs when the document is opened and when a newer version of the document has been detected
OnBeforeChange	Occurs after the data is modified but before the data bound to the node field is changed
OnValidate	Occurs after the data is modified and the data is checked against the schema
OnAfterChange	Occurs after the field data is modified, schema checks and after the node data is changed

Table 3 • Events associated with node-level data

The declarative engine is important based on the context of the user completing the form. For example, one important scenario in our new patient form was that all patient ages were greater than zero. The rules engine is accessed from the data validation tab of the field properties. It is another fill-in field dialog box that allowed us to visually build rules. If an applied rule fails, you can provide either a pop-up box or a tool tip as feedback to the user. For our scenario we felt it was better to provide the modal dialog as the preferred feedback mechanism.

InfoPath also provides an event model to allow programmatic access. In this scenario a developer can prewire events to script components that can occur within either the form's data fields or the entire form. Form-level events include form open, view switches, form submission, and data imports. At the field or node level InfoPath provides events around data change events (see Figure 3).

InfoPath provides a rich model of events that can occur both at the document and node level. In order to create your application correctly it is important to understand when these events are fired. Events associated with document lifestyle occur in the order shown in Table 2. Events associated with node-level data occur in the order shown in Table 3.

Events are created through script components that run in response to the type of trigger assigned. Currently, InfoPath supports either VB script or JavaScript components. Script language is defined at the form level using the InfoPath system options. The design of InfoPath requires that the client application is used to associate the event name, and script development be done through the Microsoft Script Editor (MSE). The event function is referred to within the form definition file, which means that you aren't able to change the function name or arguments. Figure 4 shows the form in run mode, and Figure 5 shows the architecture of how this solution would be developed.

Summary

Through this article we have developed a fully functional form for my health care customer to collect new patient information. Once collected and validated this information is auto-

Figure 4 • Form in run mode

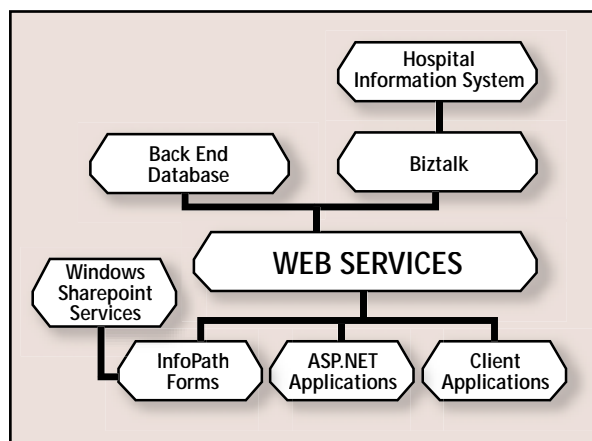


Figure 5 • Architecture

matically submitted to their back-end Web service.

Stay tuned for the second part of this series where we will extend this solution to leverage the back-end services of Biztalk Server 2002. When InfoPath is combined with this, you can create some dynamic workflow applications for your enterprise. We have covered only a very small portion of the InfoPath environment in this article. I leave it up to you to discover how InfoPath can help you reduce your "mountain of paper" and explore the further capabilities of the InfoPath 2003 application.

Resources

Several files are available for download at www.sys-con.com/xml/sourcec.cfm, including NewPatient.xsn, the actual form; Savedform.xml, a saved version of the form; and Webview.mht, a saved version of the form for the Web. 🌐

AUTHOR BIO

Thom Robbins is a senior technology specialist with Microsoft in New England. He spends his time working with customers on developing and implementing solutions with Microsoft-based technologies. He is a regular speaker at various industry events and frequent contributor to various trade magazines.



Using XML and Flash to Create a Server-Agnostic Application

XML is a key component in this ambitious project

I recently undertook an extremely ambitious project in which I decided to convert an entire application user interface from Internet Explorer to the Flash-based .swf (pronounced "swiff") format.

Flash has been reborn of late to support the development of "X" Internet applications (dubbed Rich Internet Applications by some in the industry). The small but robust Flash 6 Player has been greatly enhanced to support this new class of applications. Perhaps the most notable enhancements are its enhanced XML support and the new "components" or objects that allow sophisticated functionality to be encapsulated into a user-friendly programming metaphor. Both of these enhancements were critical for the project.

The project I worked on was a sophisticated content management application whose primary administrative user interface relied on certain "Internet Explorer-only" features including a combination of JavaScript, DHTML, CSS, and dynamically generated server-side code. The limitations of this approach went beyond the browser requirements, but that has proven to be one of the most limiting facets of the current application. Specifically, the combination of HTML, DHTML, JavaScript, CSS, and server-side code intermingled into each of the hundreds of files that make up the user interface has created a maintenance headache to say the least. What's more, this makes the code infinitely less portable across application servers – not to mention the obvious disadvantages of the coupling of server-side logic with client-side logic and presentation information.

Enter Flash and XML

The key lesson from this project was

that it's imperative to understand that any decision to completely overhaul a successful commercial shipping application is not made lightly – especially when the primary technology that will be used (Flash MX) is brand new and untested under the stresses of the anticipated workload. After several phases of "assumption" testing with some of the key processes that would be key parts of the application, it was clear that Flash could in fact handle the task, and then some. The XML object having been totally rewritten in native C++ code greatly enhanced its ability to work with large amounts of dynamically generated XML data. That, in combination with the new "component" objects metaphor, would allow form controls such as list boxes, pulldown menus, and the like to be populated from clean XML data. In addition, input elements that would typically be done as simple lists with checkboxes, for example, could now be built in a much more intuitive manner, more closely resembling that of a desktop application.

A Pleasant Surprise

Once the decision had been made to embrace the new Flash 6 runtime engine as a replacement for Internet Explorer, a fundamental and exciting development emerged. I decided that I could leverage XML to act as a configuration tool to determine the target application server that would be used to supply the data. This was significant since the previous approach made the possibility of supporting multiple application servers virtually nil. For that to happen, a completely separate code base would have to be developed and maintained. This was primarily due to the lack of decoupled code in the presentation and logic layers of the application, which was written in classic

ASP (Active Server Pages).

The new solution, as designed, would allow a single user interface to sit atop virtually any application server that could spit out XML. This was made possible with the use of a single XML file that was initially loaded into the presentation layer, which determines global paths for the data components. This XML file held the default parameters for the application. Using XML for this purpose – as opposed to a specific application server – enabled the desirable outcome of a server-agnostic application.

The XML file used is shown below:

```
<config serverProtocol="aspx" default
  Font="Tahoma" defaultFontSize="11"
  highlightColor="#FF0000" highlight
  TextColor="#FFFFFF" />.
```

In this case, the chosen application server is .NET (aspx). Consequently all data files for this application should be located under the aspx directory relative to the installation. Changing the application server then becomes a trivial matter of specifying the prefix in the serverProtocol attribute – given that a copy of the data files is available for the chosen application server.

To better understand how this all comes together, a brief explanation of the .swf runtime and its "components" feature is helpful. In the Flash runtime, user interface controls (form elements) such as text input boxes, select lists, etc., exist as components. These components can be bound to external data sources via XML using a Flash MX property called setDataProvider. In order to retrieve a data set for one of the UI components, an XML object is created in the runtime and sent to the server for the data. The server responds to the request and transfers the data back to the XML object where it is

AUTHOR BIO

Scott Blanchard is executive vice president of research and development at Octigon, a leading software development company that specializes in content management, and Web strategies and applications. The company is headquartered in Birmingham, Alabama. For more information visit www.octigon.com.

bound to the user interface control and displayed to the user for interaction. The magic part of this is how the request is built. Here is how it's done:

```
//load XML and store it in x8data
x8data= new XML();
x8data.load(_global.serverProtocol+"
    siteManager/getTemplates."+_global.
    serverProtocol+"");
x8data.onload = bindDataToList();
```

In this example, the application is asked to build a pulldown menu of the available templates in the system. To do this the path to the data file is dynamically generated using the "global.serverProtocol" variable, which holds a reference to the chosen application server protocol as it appears in the configuration file mentioned before. Using the data in the config file, the path would resolve to "aspx/siteManager/getTemplates.aspx." Thus, the only requirement for supporting multiple application servers in the application is that an application server-specific folder is available containing all the relevant data files that the runtime requires, and the application server is in fact installed and operational in order to serve the requested files to the runtime engine.

As you can see in this example, the

relevant application server file for this data transaction is "getTemplates.xxx," where "xxx" is the suffix that identifies the application server's native file format. If, on the other hand, the application server protocol was php, the path would resolve as "php/siteManager/getTemplates.php." Hence the runtime code is in no way dependent on the chosen application server.

All that is required of the chosen application server is that it return XML formatted data to the presentation layer. For example, a graphical listing of available presentation templates would be generated from the following XML data:

```
<templates>
<template id="1" name="Geneva"
    thumbnail="geneva.jpg" />
<template id="2" name="Augusta"
    thumbnail="augusta.jpg" />
<template id="3" name="St. Maarten"
    thumbnail="stmaarten.jpg" />
</templates>
```

This XML data would then be automatically bound to this particular UI component and the pulldown menu would be presented for the user. This allows the presentation layer to be cleanly separated from the application server and its code. Once server-side code has

been generated in one language, it can then be ported to another language, and so on. The beauty of the approach is that there are zero presentation instructions in the server code, creating a much greater opportunity to maintain and upgrade both the presentation layer and the logic layer, independently of one another.

Conclusion

Leveraging XML as a key component in this solution has enabled me to completely address the requirement for a server-agnostic application with an extremely simple configuration protocol. This protocol allows me to maintain a separate code base of functionally identical data files for each application server that is to be supported without ever changing the user interface files – a feat that was impossible with HTML and ASP. Extending support to future application servers is merely a question of duplicating the functionality of an existing library into the chosen language. In this way, the presentation layer has been completely separated from the logic. The same user interface code is used regardless of the chosen application server. And it's all made possible by using XML. 🌀

■ SBLANCHARD@OCTIGON.COM



PRESENTATION

Send Me Your Stylesheet

A new model allows you to customize applications and offload interface design



RSS

Revisiting RSS

Provide syndicated content to downstream apps



XMLSPY

Aggregation with XMLSPY

Best practices and practical programming techniques



STANDARDS

Human Resources Markup Language

A standard suite of XML specs to automate human resources-related data exchange

DON'T MISS XML-J JULY

REAL-WORLD SOLUTIONS



Integrating Enterprise Information on Demand with XQuery Part 1

WRITTEN BY

MICHAEL CAREY, DANIELA FLORESCU,
& NITIN MANGTANI

Addressing the EII problem

Since the dawn of the database era more than three decades ago, enterprises have been amassing an ever-increasing volume of information – both current and historical – about their operations. For the past two of those three decades, the database world has struggled with the problem of somehow integrating information that natively resides in multiple database systems or other information sources (Landers and Rosenberg).

The IT world knows this problem today as the enterprise information integration (EII) problem: enterprise applications need to be able to easily access and combine information about a given business entity from a distributed and highly varied collection of information sources. Relevant sources include various relational database systems (RDBMSs); packaged applications from vendors such as Siebel, PeopleSoft, SAP, and others; “homegrown” proprietary systems; and an increasing number of data sources that are starting to speak XML, such as XML files and Web services.

During the past two decades, a number of research and commercial systems have been built in attempts to solve the EII problem. These systems have been known by a variety of names – heterogeneous distributed database systems, multi-database systems, federated database systems, data integration systems, and now enterprise information systems. But the problem itself has persisted, and it remains a very real problem.

Solutions to the data integration problem involve choosing a common data model into which all the existing data sources are (virtually) mapped, then using a query language designed to work with that model to extract the desired data from the set of mapped data sources. Many data models and languages have been invented and/or tried over the years – including relational (SQL), functional, logical, object-oriented (ODMG/OQL), and semi-structured approaches – but each has fallen short. The two biggest impediments to their success have been the challenge of naturally mapping the data from all the sources of interest into the chosen model and the lack of industry consensus on an appropriate and acceptable model into which to map the data.

Fortunately, the XML age is upon us, and with it has come a set of technologies that are uniquely suited to solving the EII problem. Much as the simplicity of HTML and HTTP led to their rapid adoption, which in turn led to the rapid growth of the Internet, the simplicity of XML is leading to its rapid adoption as the generally accepted format for data interchange and application integration in the IT world today. Because of the rapid adoption of XML, the XML Schema standard is also rap-

idly gaining traction as the way to describe enterprise data for integration purposes. These trends are due to the simplicity and flexibility of XML – it is straightforward to express data from almost any enterprise data source in XML form without having to commit an “unnatural act.” For similar reasons, Web services – based on XML, SOAP, and WSDL – are rapidly gaining traction as the way for applications to interact, either synchronously or asynchronously, for point-to-point communication (Curbera). It follows from these trends that a query language for XML, one capable of querying and reshaping XML data as well as invoking functions, such as Web services, would provide an ideal foundation for solving the EII problem.

Enter XQuery, the emerging XML query language being produced by the W3C XML Query working group. In this article, we provide an introduction to XQuery and explain how it enables true enterprise information integration – allowing not just database data, but also information from applications, Web services, messages, XML files, and other data sources, to be integrated into coherent reusable views and then used to meet the query demands of enterprise applications.

XQuery: A Query Language for XML

The development of the SQL language for querying and manipulating relational data was a major force in ushering in the database age in the late 1970s and early 1980s. The goal of the W3C XML Query working group has been to design a similarly high-level, declarative query language for XML data.

Why not SQL?

It's natural to ask why SQL, or a SQL derivative, isn't the right solution to the problem of querying XML. The answer is that there are just too many differences between XML data and relational data to make SQL a good candidate for this task:

- Relational tables are flat, whereas XML data tends to be hierarchically structured, often several levels deep.
- Relational tables are highly uniform, while XML data tends to be more highly variable. Structural variations, typing variations, and missing data are more the norm than the exception with XML data.
- Relational data is naturally unordered, while order often has an important meaning in XML data (particularly for document data!).
- Tables have relatively static schemas that can be difficult to evolve, while XML Schemas tend to be more extensible, and the self-describing nature of XML blurs the data/meta-data distinction. Moreover, XML data may or may not have an associated schema, while relational data cannot exist in the absence of a schema.

- Finally, in the XML world, textual information can be inter-mixed freely with structured (i.e., tagged) information.

As a result, the W3C has been designing a new query language tailored to the unique needs of manipulating XML data. The result of that work is the language now called XQuery. Although XQuery is a work in progress, it is nearing completion at the time of this writing, and it is likely to become an official W3C Recommendation in late 2003.

XQuery basics

At the heart of the semantics of XQuery, and also of XPath 2.0, lies the XQuery data model. Just as the relational model laid the foundation for SQL, the XQuery data model lays the foundation for XQuery. Because XML data is naturally ordered, the XML data model is based on the notion of ordered trees. Central to the XML data model is the notion of a sequence. XML queries consume and produce sequences that consist of atomic values (based on the primitive types of XML Schema) and/or of XML nodes (element, attribute, text, document, and so on).

XQuery is a functional, side-effect-free language. Like many other functional languages, a program (a query in the case of XQuery) consists of a prologue and a body, where the body is an expression. The result of a given query is the result of evaluating its body expression in the environment defined by its prologue. Expressions in XQuery can be simple expressions like primitive constants (e.g., "John Doe" or 1.3), variables, arithmetic expressions, function calls, or path expressions (familiar to users of XPath). They can also be combined to form more interesting expressions via operators, functions, and syntactic constructs including FLWOR expressions (discussed shortly), typeswitch expressions, and node constructors.

The XQuery language is rich enough to support navigation within an XML input document, the combining of data from multiple XML inputs, and the generation of new XML structures from one or more XML inputs. To generate new XML structures, XQuery takes a JSP-like approach. A subset of the XML syntax itself is part of the XQuery language, enriched with XQuery expressions that are executed dynamically and replaced inside the XML structures with their results. One can switch between literal XML and query expressions via curly braces.

From the standpoint of the EII problem, the most important expression in XQuery is the FLWOR (pronounced "flower") expression, which is roughly analogous to SELECT-FROM-WHERE-ORDER BY queries in SQL. The components of a FLWOR expression are:

- A for clause that generates one or more value sequences, binding the values to query variables. The for clause in XQuery plays a role similar to the FROM clause in SQL.
- A let clause that binds a temporary variable to the result of a query expression. The XQuery let clause is similar to support for temporary views in some dialects of SQL.
- A where clause that contains Boolean predicates that restrict the FOR clause's variable bindings. The where clause in XQuery serves the same purpose as the WHERE clause in SQL.
- An order by clause that contains a list of expressions that dictate the order of the FLWOR expression's XML output. XQuery's order by clause is directly analogous to SQL's ORDER BY clause.
- A return clause that specifies the query's desired XML output. The XQuery return clause is analogous to the SELECT clause in SQL, but the structures that it can specify are

much richer than those expressible in SQL. (For example, this is where XQuery's JSP-like XML node construction syntax can be found.)

For data handling, XQuery has the richness of SQL and more – XQuery includes support for subqueries, union, intersection, difference, aggregate functions, sorting, existential and universal quantification, conditional expressions, user-defined functions (that may even be recursive), and static and dynamic typing, in addition to various constructs to support document manipulation (e.g., query primitives for order-related operations). The biggest thing that XQuery lacks relative to SQL today is support for updates; XQuery 1.0 is strictly a functional data access language, with update support being targeted for a later revision of the standard.

Using XQuery for Enterprise Information Integration

To show how XQuery can be applied to solve the EII problem, as well as illustrate the power of some of the main constructs of the language, let's consider a simple yet illustrative business scenario. A large consumer electronics retailer wants to organize its IT infrastructure to make its staff more productive and its business more effective. The retailer has both in-store customers and online customers, and it both sells and services home entertainment systems, computers, and other consumer electronic devices. To encourage customer loyalty, consumers receive reward points for their purchases.

The bottom layer of Figure 1 shows what the electronics retailer's IT infrastructure looks like today. Its customer relationship management (CRM) data, such as information about customers and credit cards, is stored in an RDBMS. Order management is handled through an ERP system (SAP), and as a result, order information is available via a J2EE-CA adapter developed to access the ERP system's API. The adapter API

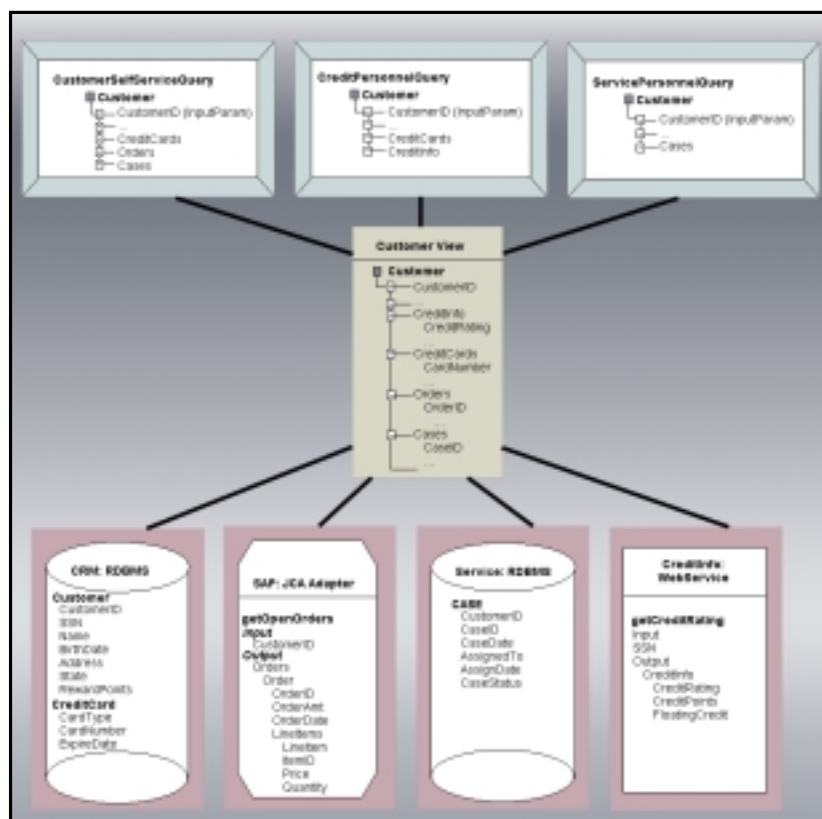


Figure 1 • Customer data integration example

provides calls like `getOpenOrders()`, which takes a customer ID as input and returns a list of that customer's open order information. Service data is also stored in an RDBMS, but in a different one than the customer data. Finally, the electronics retailer utilizes an external service for performing customer credit checks. The external credit service provides a `getCreditRating()` Web service call that takes a social security number – formatted differently than in the electronic retailer's RDBMS – and runs a credit check on the specified individual.

The electronics retailer's line of business managers have asked the company's IT department to create customer portals for three different sets of users. The three desired portals and their data provision requirements are:

- An online customer self-service portal that will be directly accessed by customers via the Internet. This customer self-service portal should show the customer's profile information, registered credit cards, orders, and service cases, but it should not show the customer's credit rating information.
- A credit approval portal that will only be accessed by credit approval personnel. This credit approval portal should show the customer's basic profile information, registered credit cards, and credit rating information.
- An internal product service portal that will be used by clerks in the electronic retailer's service department. This service portal should show just the customer's basic profile information and service case information.

“XQuery is a functional, side-effect-free language”

All three portals require information about the same core business entity – namely the customer. However, each line of business manager wants a different view of the customer. The electronics retailer's IT department wanted a solution that would enable rapid development and provide high reusability of their initial data integration efforts as well as subsequent low maintenance. Fortunately, their data architects realized that they could achieve these goals by creating a single, integrated base view of the customer and then creating three application-specific views on top of the base view. This way, their data integration effort is spent on creating the base view, and the application-specific views are then easily created without concern for disparate data models, differing data source APIs, or other integration snafus. Later on, changes in underlying data source schemas can be dealt with by maintaining the base view; the application-specific views are shielded from most such changes.

With XQuery, the solution sketched above can be implemented by viewing the enterprise's different data sources all as virtual XML documents and functions. XQuery can stitch the distributed customer information together into a comprehensive, reusable base view. That is, the base view definition can be expressed using XQuery, respecting the hierarchical nature of the data, given appropriate default XML views of the enterprise's data sources. As indicated in Figure 1, the relational data sources can be exposed using simple default XML Schemas, and the other sources – SAP and the credit-checking Web service – can be exposed to XQuery as callable XQuery functions with appropriate signatures. In the middle of Figure 1, we see a sketch of the desired “single view of customer” –

here, the desire is for all data about customers to be made available for easy querying from various applications. The developers of these applications then simply work against this unified view – which is an XML view of customers where each customer has some basic data, some credit rating information, an associated set of credit cards, a set of open orders (each with all their line item details nested inside), and a set of service cases.

Listing 1 shows in full detail how XQuery can be used to define the desired single view of customer. The XQuery shown in the listing defines a single well-formed XML document with top-level element `CUSTPROFILE`. The outermost `FLWOR` expression uses the variable `$Cust` to “iterate” logically over all of the customers in the CRM database's `CUST` table. Its `let`-clause binds a second variable, `$CredInfo`, to the result of calling the credit Web service's method `getCreditRating()`. Note that this call deals with the disparate social security number formats by reformatting the value being passed to the Web service. The top-level `return`-clause is where most of the action is, as this is where the desired result is defined and shaped. For each customer, the view will contain a `CUST` element with the basic customer data at the top level and a nested `CREDITINFO` element with the customer's credit rating from the Web service. It will have a `CREDITCARDS` element containing subelements for each of the customer's credit cards, computed via a correlated `FLWOR` subquery (much like a nested query in SQL), and the view query has similar subqueries for computing the sets of `ORDERS` and `CASES` for each customer. In the case of `ORDERS`, notice that the subquery's `for`-clause ranges over the result obtained by calling the `getOpenOrders()` method of the ERP application adapter. Like the Web services call, this method appears to the view definer as another callable XQuery function.

As shown at the top of Figure 1, there are three different queries to be written against the base customer view. One is `CustomerSelfServiceQuery`, a parameterized query that, given a customer ID, returns the information that the customer is allowed to see through the customer self-service portal. This query returns everything known about the customer except for the `CREDITINFO` element. Another query is `CreditPersonnelQuery`, for use by the personnel who handle credit approval requests. This query also takes a customer ID and returns customer information; however, it omits `ORDERS` and `CASES`, as they are not relevant for credit department use. The third query in Figure 1, `ServicePersonnelQuery`, is for use by the service department. This query takes a customer ID and returns basic information about the customer plus the set of open service cases for the customer. Listing 2 shows how simple it is to write the third query given the centralized customer view provided by Listing 1.

This example, while it uses very simple data sources and schemas for clarity, illustrates a number of important points about the benefits of an XQuery-based EII solution. One benefit is that the data integration problem for a given business entity only needs to be solved once, when defining the centralized view. It can then be leveraged across multiple applications, and the queries or further views for those applications are vastly simplified (as shown by Listing 2). Another benefit is that the use of XML and XQuery provide a very natural basis for defining centralized views of real enterprise data. They make it simple to capture the naturally hierarchical nature of the data, particularly for data that lives within applications (as opposed to just flat RDBMS tables). XQuery also provides the power to deal with complications like key mismatches, either by calling a function to transform a key, as is done in the Web

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

AND SAVE UP TO \$400 AND RECEIVE UP TO 3 FREE CDs!

RECEIVE
YOUR DIGITAL
EDITION
ACCESS CODE
INSTANTLY
WITH YOUR PAID
SUBSCRIPTIONS

3-Pack

Pick any 3 of our magazines and save up to **\$275⁰⁰**
Pay only \$175 for a 1 year subscription plus a **FREE CD**

- 2 Year - \$299.00
- Canada/Mexico - \$245.00
- International - \$315.00

6-Pack

Pick any 6 of our magazines and save up to **\$350⁰⁰**
Pay only \$395 for a 1 year subscription plus **2 FREE CDs**

- 2 Year - \$669.00
- Canada/Mexico - \$555.00
- International - \$710.00

9-Pack

Pick 9 of our magazines and save up to **\$400⁰⁰**
Pay only \$495 for a 1 year subscription plus **3 FREE CDs**

- 2 Year - \$839.00
- Canada/Mexico - \$695.00
- International - \$890.00



TO
ORDER

• Choose the Multi-Pack you want to order by checking next to it below. • Check the number of years you want to order. • Indicate your location by checking either U.S., Canada/Mexico or International. • Then choose which magazines you want to include with your Multi-Pack order.

Pick a 3-Pack, a 6-Pack or a 9-Pack

<input type="checkbox"/> 3-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.
<input type="checkbox"/> 6-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.
<input type="checkbox"/> 9-Pack	<input type="checkbox"/> 1YR <input type="checkbox"/> 2YR	<input type="checkbox"/> U.S. <input type="checkbox"/> Can/Mex <input type="checkbox"/> Intl.

☐ Linux Business & Technology

U.S. - Two Years (24) Cover: \$143	You Pay: \$79.99 /	Save: \$63 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$39.99 /	Save: \$32
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$79.99 /	Save: \$4
Intl - Two Years (24) \$216	You Pay: \$176 /	Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ Java Developer's Journal

U.S. - Two Years (24) Cover: \$144	You Pay: \$89 /	Save: \$55 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$49.99 /	Save: \$22
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$79.99 /	Save: \$4
Intl - Two Years (24) \$216	You Pay: \$176 /	Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ Web Services Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ .NET Developer's Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ XML-Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ WebLogic Developer's Journal

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

☐ ColdFusion Developer's Journal

U.S. - Two Years (24) Cover: \$216	You Pay: \$129 /	Save: \$87 + FREE \$198 CD
U.S. - One Year (12) Cover: \$108	You Pay: \$89.99 /	Save: \$18
Can/Mex - Two Years (24) \$240	You Pay: \$159.99 /	Save: \$80 + FREE \$198 CD
Can/Mex - One Year (12) \$120	You Pay: \$99.99 /	Save: \$20
Intl - Two Years (24) \$264	You Pay: \$189 /	Save: \$75 + FREE \$198 CD
Intl - One Year (12) \$132	You Pay: \$129.99 /	Save: \$2

☐ Wireless Business & Technology

U.S. - Two Years (24) Cover: \$144	You Pay: \$89 /	Save: \$55 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$49.99 /	Save: \$22
Can/Mex - Two Years (24) \$192	You Pay: \$139 /	Save: \$53 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$79.99 /	Save: \$16
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ WebSphere Developer's Journal

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

☐ PowerBuilder Developer's Journal

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Subscribe Online Today www.sys-con.com/2001/sub.cfm

SYS-CON
MEDIA

service call in Listing 1, or by incorporating a key mapping table or service into the base view query.

It is important to mention that these benefits come with no requisite negative performance implications. When the XQuery-based EII system goes to process a query like the one in Listing 2, it will do inline-like expansion of the query's view reference (as has been done for decades in RDBMSs). This will result in a query that involves only the base data sources, a query in which predicates such as the customer ID parameter and the "Open" case status constant can be pushed all the way down to the appropriate data sources. Also, only those base sources that actually contain data needed for the query – the two RDBMSs in Listing 2's case, for example – will become involved in processing the query at runtime.

In Part 1 of this article we have introduced the EII problem, provided a brief overview of XQuery, and explained XQuery's role in solving the EII problem. In Part 2, we will complete the picture by talking about two related technologies, namely EAI and ETL, and explaining how they relate to EII and XQuery. We will also describe an EII customer scenario and explain how Liquid Data for WebLogic, an XQuery-based EII product from BEA, was used to tackle the data integration problems that this customer faced. ☛

References

- Landers, T., and Rosenberg, R., "An Overview of Multibase."

Proceedings of the 2nd International Symposium on Distributed Data Bases, Berlin, Germany. North-Holland Publishing Co., September 1982.

- Curbera, E., et al. "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI." *IEEE Internet Computing* 6(2), March-April 2002.
- XQuery 1.0: www.w3.org/TR/xquery

AUTHOR BIOS

Michael Carey is the architect for Liquid Data for WebLogic at BEA Systems. His previous jobs include five years at IBM working on various DB2-related projects and 12 years as a professor at the University of Wisconsin-Madison. He is a recognized expert on database system architectures and performance evaluation.

Daniela Florescu is an editor for the W3C XQuery language. She works at BEA Systems in the Liquid Data team. She was the architect of the XQuery query processor used for the data transformation engine of WebLogic Integration 8.1. and she is widely known for her past work on data integration and query processing.

Nitin Mangtani is currently the technical program manager for Liquid Data for WebLogic at BEA Systems. As a founding member of the Liquid Data team, Nitin has been instrumental in defining the overall product vision as well as shipping Liquid Data. Prior to joining BEA, he worked on distributed order management systems at I2 Technologies.

	MCAREY@BEA.COM
	DANIELAF@BEA.COM
	NITINM@BEA.COM

LISTING 1 • Single view of customer

```
<CUSTPROFILE>
{
  for $Cust in document("BEACRM")/db/CUST
  let $CredInfo := WSCredit:getCredInfo(replace($Cust/
    SSN, "-", ""))/CREDIT
  return
    <CUST>
      <CUSTID>{ data($Cust/CUST_ID) }</CUSTID>
      <NAME>{ data($Cust/FULL_NAME) }</NAME>
      <BIRTHDATE>{ data($Cust/BIRTH_DATE) }</BIRTHDATE>
      <SSN>{ data($Cust/ SSN) }</SSN>
      <ADDRESS>{ data($Cust/STREET_ADDRESS) }</ADDRESS>
      <STATE>{ data($Cust/STATE) }</STATE>
      <REWARDPOINTS>{ data($Cust/R_POINTS) }</REWARD-
POINTS>
      <CREDITINFO>
        <RATING>{ data($CredInfo/CREDIT_RATING) }</RATING>
        <POINTS>{ data($CredInfo/CREDIT_POINTS) }</POINTS>
        <CREDIT>{ data($CredInfo/FLOAT_CREDIT) }</CREDIT>
      </CREDITINFO>
      <CREDITCARDS>
      {
        for $CredCard in document("BEACRM")/db/CREDIT_CARD
        where ($Cust/CUST_ID eq $CredCard/CUST_ID)
        return
          <CREDITCARD>
            <TYPE>{ data($CredCard/CARD_TYPE) }</TYPE>
            <NUMBER>{ data($CredCard/CARD_NUMBER) }</NUMBER>
            <EXPDATE>{ data($CredCard/EXPIRY_DATE) }
            </EXPDATE>
          </CREDITCARD>
      }
      </CREDITCARDS>
      <ORDERS>
      {
        for $CustOrder in
          OrderERP:getCUSTOrder($Cust/CUST_ID)/ORDER
        return
          <ORDER>
            <ORDERID>{data($CustOrder/ORDER_ID) }</ORDERID>
            <ORDERDATE>{data($CustOrder/ORDER_DATE) }
            </ORDERDATE>
            <ORDERAMT>{data($CustOrder/ORDER_AMT) }</ORDER-
AMT>
            <STATUS>{data($CustOrder/ORDER_STATUS) }</STATUS>
            {
```

```
for $LineItem in $CustOrder/LINE_ITEM
return
  <LINE_ITEM>
    <ITEMID>{data($LineItem/ITEM_CODE) }</ITEMID>
    <PRICE>{data($LineItem/PRICE) }</PRICE>
    <QTY>{data($LineItem/QUANTITY) }</QTY>
    <STATUS>{data($LineItem/STATUS) }</STATUS>
  </LINE_ITEM>
}
</ORDER>
}
</ORDERS>
<CASES>
{
  for $Case in document("BEASERVICE")/db/CASE
  where ($Case/CUST_ID eq $Cust/CUST_ID)
  return
    <CASE>
      <CASEID>{ data($Case/CASE_ID) }</CASEID>
      <OPEN_DATE>{ data($Case/CASE_DATE) }</OPEN_DATE>
      <ASSIGNTO>{ data($Case/ASSIGN_TO) }</ASSIGNTO>
      <STATUS>{ data($Case/STATUS) }</STATUS>
    </CASE>
  }</CASES>
</CUST>
}
</CUSTPROFILE>
```

LISTING 2 • Service personnel query

```
define variable $CustID as xs:string external

<CUSTSVCPROFILE>
{
  for $Cust in view:CustomerProfile/CUST
  where $Cust eq $CustID return
    <CUST>
      <CUSTID>{ data($Cust/CUST_ID) }</CUSTID>
      <NAME>{ data($Cust/FULL_NAME) }</NAME>
      <ADDRESS>{ data($Cust/STREET_ADDRESS) }</ADDRESS>
      <STATE>{ data($Cust/STATE) }</STATE>
      <OPENCASES>
      {
        $Cust/CASES/CASE [STATUS eq "Open"] }
      </OPENCASES>
    </CUST>
  }
</CUSTSVCPROFILE>
```

Download the Code
www.sys-con.com/xml

International Web Services Conference & Expo

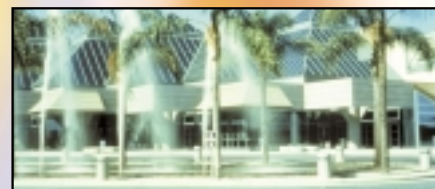
Web Services Edge ^{WEST} 2003

web services
conference & expo

EDGE
conference & expo

SEPT. 30 - OCT. 2, 2003

Santa Clara, CA



**EXTENDING THE ENTERPRISE
WITH WEB SERVICES THROUGH JAVA,
.NET, WEBSHERE, MAC OS X
AND XML TECHNOLOGIES**



XML

WebSphere

Microsoft
.net



Mac OS X



LINUX EDGE
conference & expo

web services
conference & expo

BOSTON

February 24-27, 2004

- Featured technologies and topics will include:
- Focus on XML
 - Focus on .NET
 - Focus on Java
 - Focus on WebSphere
 - Focus on Mac OS X

For more information visit
www.sys-con.com
or call
201 802-3069



Over 100 participating companies will display and demonstrate over 300 developer products and solutions.

Over 2,000 Systems Integrators, System Architects, Developers, and Project Managers will attend the conference expo.

Over 60 of the latest sessions on training, certifications, seminars, case studies, and panel discussions will deliver REAL World benefits, the industry pulse and proven strategies.

Contact information: U.S. Events: 201 802-3069 or e-mail grisha@sys-con.com

WebServices
JOURNAL

JAVA
DEVELOPER'S JOURNAL

WebSphere
DEVELOPER'S JOURNAL

XML
JOURNAL

.NET
DEVELOPER'S JOURNAL

SAMS

WebLogic
DEVELOPER'S JOURNAL

wireless
BUSINESS TECHNOLOGY

LINUX
BUSINESS TECHNOLOGY

CE Advisor

ColdFusion
DEVELOPER'S JOURNAL

PowerBuilder
DEVELOPER'S JOURNAL

Java is a registered trademark of Sun Microsystems. .NET is a registered trademark of Microsoft. Mac OS X is a registered trademark of Apple Computer, Inc. WebSphere is a registered trademark of IBM. All other product names herein are the properties of their respective companies.

**WEB SERVICES EDGE WEST 2003
CALL FOR PAPERS NOW OPEN**

Submit your papers online at:
www.sys-con.com/webservices2003west

PRODUCED BY
sys-con
EVENTS



XML Certification Quizzer

A review of some key concepts

Last month's column introduced you to the exam objectives for IBM Test 141 on XML and Related Technologies. This installment will present five more questions from the five different exam objectives. It can be used as a learning tool for people who are new to XML. If you are preparing for the exam, the article will help you review key concepts.

Question 1 (XML Processing)

Which of the following XSLT elements sets the value of a variable *x* to the literal string value "a"?

- A. `<xsl:variable name='x' select='a' />`
- B. `<xsl:variable name="x" value="a"/>`
- C. `<xsl:variable name='x' select="'a'"/>`
- D. `<xsl:variable select="'a'" value="x" />`

Select one answer.

Explanation: Choice C is the correct answer. There are three methods for specifying the value of a variable:

1. Use an empty `xsl:variable` element with a select attribute. The value of the variable is the object that results from evaluating the expression in the select attribute.
2. Use a nonempty `xsl:variable` element with no select attribute. In this case, the `xsl:variable` has one or more child nodes, which form a template. The template is instantiated to produce a result tree fragment, which is the value of the variable. The root node of the result tree fragment contains a sequence of child nodes generated by instantiating the template.
3. Set the value of the variable to an empty string by using an empty `xsl:variable` element with no select attribute.

In order to assign a literal string value to a variable, the value must be delimited with quotes. These quotes must be placed within the quotes used to delimit the value of the select attribute. The following will set the value of a variable *x* to the literal string value "a":

```
<xsl:variable name='x' select="'a'"/>
<xsl:variable select="'a'" name="x"/>
```

Note that in XML, the ordering of attributes is not important and attribute values can be delimited either by a pair of single quotes or a pair of double quotes.

Choice A is not correct, because it sets the value of the variable *x* to a node set containing all the elements `<a>`, children of the context node. The XSLT stylesheet may not generate an error, but will certainly produce an output that is different from the expected result.

Choices B and D are not correct because XSLT does not specify the value attribute on the `xsl:variable` element.

Question 2 (Information Modeling)

Consider the following DTD declarations:

```
<!ELEMENT   CrossReference (#PCDATA)>
<!ATTLIST  CrossReference
Linkend    IDREF    #REQUIRED>
```

Which of the following XML Schema declarations is the best translation of the DTD declarations above?

- A. `<xsd:element name = "CrossReference">`
`<xsd:complexType>`
`<xsd:simpleContent>`
`<xsd:extension base =`

```
"xsd:string">
<xsd:attribute name = "Linkend"
use = "required" type = "xsd:IDREF"/>
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
</xsd:element>
```

- B. `<xsd:element name = "CrossReference"`
`type="xsd:string">`
`<xsd:attribute name = "Linkend" use =`
`"required" type = "xsd:IDREF"/>`
`</xsd:element>`

- C. `<xsd:element name = "CrossReference">`
`<xsd:complexContent>`
`<xsd:extension base = "xsd:string">`
`<xsd:attribute name = "Linkend"`
`use = "required"`
`type = "xsd:IDREF"/>`
`</xsd:extension>`
`</xsd:complexContent>`
`</xsd:element>`

Select one answer.

Explanation: Choice A is the correct answer. The `CrossReference` element has an attribute. Simple types cannot have attributes. Therefore, the declaration of the `CrossReference` element must contain a complex type definition.

The content model of `CrossReference` contains only character data and no child elements. The `simpleContent` element is used inside the `complexType` to indicate that `CrossReference` is an extension of the simple type `xsd:string`. Since the simple type `xsd:string` is extended by adding a `Linkend` attribute, we use an extension element inside `simpleContent`.

Choice B is not correct. The element declaration does not specify whether `CrossReference` is a simple type or a

AUTHOR BIO

Joel Amoussou is founder and chief learning architect of XMLMentor.Net, where he develops blended learning solutions for building and assessing XML skills. Joel is the author of an XML exam simulator and teaches live e-learning courses on XML certification.

complex type.

Choice C is not correct. The `complexContent` element is used to restrict or extend the content model of a complex type.

Question 3 (XML Rendering)

Which of the following XSL-FO fragments will display the content of pages in two columns?

A.

```
<fo:simple-page-master master-name="one"
    page-height="11in"
    page-width="8.5in"
    margin-top="1in"
    margin-bottom="1in"
    margin-left="0.75in"
    margin-right="0.75in">
  <fo:column
    margin-top="1in" margin-bottom="1in"
    column-count="2" column-gap="0.25in"/>
  <fo:region-before extent="1in" />
  <fo:region-after extent="1in" />
</fo:simple-page-master>
```

B.

```
<fo:simple-page-master master-name="one"
    page-height="11in"
    page-width="8.5in"
    margin-top="1in"
    margin-bottom="1in"
    margin-left="0.75in"
    margin-right="0.75in">
  <fo:region-body
    margin-top="1in" margin-bottom="1in"
    column-count="2" column-gap="0.25in"/>
  <fo:region-before extent="1in" />
  <fo:region-after extent="1in" />
</fo:simple-page-master>
```

C.

```
<fo:page-sequence master-name="one"
    page-height="11in"
    page-width="8.5in"
    margin-top="1in"
    margin-bottom="1in"
    margin-left="0.75in"
    margin-right="0.75in">
  <fo:region-body
    margin-top="1in" margin-bottom="1in"
    column-count="2" column-gap="0.25in"/>
  <fo:region-before extent="1in" />
  <fo:region-after extent="1in" />
</fo:page-sequence>
```

Select one answer.

Explanation: Choice B is the correct answer. The `fo:simple-page-master` is used to specify the dimensions of the page (page-height and page-width) and margin properties, including margin-top, margin-bottom, margin-left, and margin-right. In addition, the `fo:simple-page-master` specifies the five regions of the page: region-body, region-before, region-after, region-start, and region-

end.

If the page is subdivided in multiple columns, a `column-count` attribute is attached to the `fo:region-body` element to provide multiple columns. The `column-gap` attribute specifies the width of the separation between adjacent columns.

The `fo:page-sequence` formatting object specifies a sequence or sub-sequence of pages within a document. Its `fo:flow` and `fo:static-content` flow objects child elements specify the content of these pages.

Choice A is not correct because there is no `fo:column` formatting object defined in XSL.

Choice C is not correct because the allowed content of `fo:page-sequence` is the following: (title?,static-content*, flow).

Question 4 (Testing & Tuning)

Consider the following complex type definitions:

```
<xsd:schema xmlns:xsd="http://www.w3.org
/2001/XMLSchema"
  xmlns:di="http://www.xdrugs.com/di"
  targetNamespace="http://www.xdrugs.com/
di">
```

```
<xsd:complexType name="PatientInfo">
  <xsd:sequence>
    <xsd:element name="patientid"
      type="xsd:string"/>
    <xsd:element name="name"
      type="xsd:string"/>
    <xsd:element name="illness"
      type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="DrugInfo">
  <xsd:sequence>
    <xsd:element name="id"
      type="xsd:string"/>
    <xsd:element name="dosage"
      type="xsd:string"/>
    <xsd:element name="consumer"
      type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

```
</xsd:schema>
```

A prescriptions element must be defined in the schema. Prescriptions elements must contain one or more patient elements of type `PatientInfo` followed by one or more drug elements of type `DrugInfo`. Inside a prescriptions element, the `patientid` child of patient must be unique, and each consumer

child of drug must have a corresponding `patientid`.

Which of the following is the BEST method for modeling the content of the prescriptions element?

A.

```
<xsd:element name="prescriptions">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="patient" type="di:
        PatientInfo"
        minOccurs="1" maxOccurs="unbounded"
        type="xsd:ID"/>
      <xsd:element name="drug" type=
        "di:DrugInfo" minOccurs="1"
        maxOccurs="unbounded"
        type="xsd:IDREF"/>
    </xsd:sequence>
  </xsd:complexType>
```

B.

```
<xsd:element name="prescriptions">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="patient" type="di:
        PatientInfo"
        minOccurs="1" maxOccurs="unbounded"/>
      <xsd:element name="drug" type="di:
        DrugInfo" minOccurs="1" maxOccurs="
        unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
```

```
<xsd:ID name='PatientKey'>
  <xsd:selector xpath='patient'/>
  <xsd:field xpath='patientid'/>
</xsd:ID>

<xsd:IDREF name='DrugKeyRef' refer='di:
  PatientKey'>
  <xsd:selector xpath='drug'/>
  <xsd:field xpath='consumer'/>
</xsd:IDREF>
</xsd:element>
```

C.

```
<xsd:element name="prescriptions">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="patient" type="di:
        PatientInfo"
        minOccurs="1" maxOccurs="unbounded"/>
      <xsd:element name="drug" type="di:Drug
        Info" minOccurs="1" maxOccurs=
        "unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
```

```
<xsd:key name='PatientKey'>
  <xsd:selector xpath='patient'/>
  <xsd:field xpath='patientid'/>
</xsd:key>
```

```
<xsd:keyref name='DrugKeyRef'
  refer='di:PatientKey'>
  <xsd:selector xpath='drug'/>
  <xsd:field xpath='consumer'/>
```

</xsd:keyref>
</xsd:element>

Select one answer.

Explanation: Choice C is the correct answer. The key element applies to the content of patientid elements that are children of the patient element. This means that the value of patientid must be unique and is not nillable. The key can be referenced from elsewhere with its name, PatientKey. To ensure that consumer children of the drug elements have a corresponding value in the patientid elements identified by the key, we use the keyref element.

Choice A and Choice B are not correct. For compatibility with XML1.0, ID and IDREF should be used only on attributes.

Question 5 (Architecture)

An international industry group of airlines, aircraft manufacturers, and parts suppliers decides to adopt Web services for the parts ordering process. Airlines must be able to query thousands of manufacturer and supplier parts catalogs and place orders based on availability, price, quantity, quality, and

shipping costs. The industry group forms a Web Services Working Group to provide technical recommendations on the adoption of Web services by its members.

Which of the following is *least* likely to be part of the recommendations of the Working Group?

- A. The industry group must define a WSDL service interface definition, and register it as a tModel in a UDDI registry.
- B. SOAP request and response messages must be exchanged over HTTP.
- C. The industry group must create an XML Schema to describe the messages used to search aircraft parts catalogs, place orders, and track orders.
- D. Each aircraft manufacturer or part supplier must define its own specific WSDL service interface definition, and put a copy of the WSDL file on its Web server.

Select one answer.

Explanation: Choice D is the correct answer. If each aircraft manufacturer or part supplier defines its own specific WSDL service interface definition, an

airline must know the service provider in advance before it can obtain the WSDL definition. Since there are thousands of manufacturers and suppliers, that solution will not bring significant efficiency gains.

Once an industry standard WSDL service interface definition is registered as a tModel in a UDDI registry, programmers at airlines can retrieve the WSDL document by following the overviewDoc link in the tModel.

The network address of each service access point is encoded in the accessPoint element of a bindingTemplate. Client applications will query the UDDI registry for the access points of all registered Web services that implement the industry standard WSDL.

Conclusion

When preparing for IBM XML Certification, it is important to understand how various XML-related specifications are used together to design a solution, and the role of each specification in the architecture. ☺

JOEL@XMLMENTOR.NET

XML-J ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Altova	http://xmij.altova.com/authentic5	978-816-1600	36
Ektron	www.ektron.com/xmij	603-594-0249	17
IBM	ibm.com/websphere/seeit		4
JavaOne	java.sun.com/javaone/sf		11
Linux Business & Technology	www.sys-con.com	888-303-5282	33
Macromedia	www.macromedia.com/go/cfmxad	415-252-2000	2
Mindreef	www.mindreef.com	603-465-2204	9
Quest Software, Inc.	http://java.quest.com/jcsc/ws	800-663-4723	35
SYS-CON Media	www.sys-con.com/2001/sub.cfm	888-303-5282	27
SYS-CON Reprints	www.sys-con.com	201-802-3026	32
Web Services Edge West 2003	www.sys-con.com	201-802-3069	29

General Conditions: The Publisher reserves the right to refuse any advertising not meeting the standards that are set to protect the high editorial quality of *XML-Journal*. All advertising is subject to approval by the Publisher. The Publisher assumes no liability for any costs or damages incurred if for any reason the Publisher fails to publish an advertisement. In no event shall the Publisher be liable for any costs or damages in excess of the cost of the advertisement as a result of a mistake in the advertisement or for any other reason. The Advertiser is fully responsible for all financial liability and terms of the contract executed by the agents or agencies who are acting on behalf of the Advertiser. Conditions set in this document (except the rates) are subject to change by the Publisher without notice. No conditions other than those set forth in this "General Conditions Document" shall be binding upon the Publisher. Advertisers (and their agencies) are fully responsible for the content of their advertisements printed in *XML-Journal*. Advertisements are to be printed at the discretion of the Publisher. This discretion includes the positioning of the advertisement, except for "preferred positions" described in the rate table. Cancellations and changes to advertisements must be made in writing before the closing date. "Publisher" in this "General Conditions Document" refers to SYS-CON Publications, Inc. This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.

Once you're in it...



reprint it...



Contact Carrie Gebert
201 802-3026
carrieg@sys-con.com

SYS-CON MEDIA

Re Prints

Premiering
June 2003
at
JavaOne

www.sys-con.com



Millions of Linux Users One Magazine

Linux Business & Technology

There is no escaping the penetration of Linux into the corporate world. Traditional models are being turned on their head as the open-for-everyone Linux bandwagon rolls forward.

Linux is an operating system that is traditionally held in the highest esteem by the hardcore or geek developers of the world. With its roots firmly seeded in the open-source model, Linux is very much born from the "if it's broke, then fix it yourself" attitude.

Major corporations including IBM, Oracle, Sun, and Dell have all committed significant resources and money to ensure their strategy for the future involves Linux. Linux has arrived at the boardroom.

Yet until now, no title has existed that explicitly addresses this new hunger for information from the corporate arena. *Linux Business & Technology* is aimed squarely at providing this group with the knowledge and background that will allow them to make decisions to utilize the Linux operating system.

Look for all the strategic information required to better inform the community on how powerful an alternative Linux can be. *Linux Business & Technology* will not feature low-level code snippets but will focus instead on the higher logistical level, providing advice on hardware, to software, through to the recruiting of trained personnel required to successfully deploy a Linux-based solution. Each month will see a different focus, allowing a detailed analysis of all the components that make up the greater Linux landscape.

Regular features will include:

- Advice on Linux Infrastructure
- Detailed Software Reviews
- Migration Advice
- Hardware Advice
- CEO Guest Editorials
- Recruiting/Certification Advice
- Latest News That Matters
- Case Studies

LINUX EDGE
conference & expo

June 3-5 LONDON
June 24-26 BERLIN
September HONG KONG
October CALIFORNIA

**SYS-CON
MEDIA**

The World's Leading i-Technology Publisher

SAVE 30% OFF!

REGULAR ANNUAL COVER PRICE \$71.76

YOU PAY ONLY

\$49.99

12 ISSUES/YR

*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

SUBSCRIBE TODAY!

WWW.SYS-CON.COM

OR CALL

1-888-303-5282

FOR ADVERTISING INFORMATION:

CALL 201.802.3020 OR

VISIT WWW.SYS-CON.COM

ALL BRAND AND PRODUCT NAMES USED ON THIS PAGE ARE TRADE NAMES, SERVICE MARKS, OR TRADEMARKS OF THEIR RESPECTIVE COMPANIES.

Appligent Releases AppendPDF Pro 3.0

(Lansdowne, PA) – Appligent, Inc., a leading provider of PDF-related software solutions, has announced the release of AppendPDF Pro 3.0, which enables businesses and organizations to dynamically assemble sections from PDF documents to build a completely new version with a choice of personalized features, such as a cover page and table of contents.



This allows any PDF file to be automatically built and customized to the needs of each individual requesting specific information.

New features of AppendPDF Pro include support for using the popular XML format—an industry standard for structured content exchange used by many IT departments—to create parameter files, which control the operation of the application. The XML standard mandates that the structure of the content be validated when it is created, virtually eliminating syntax errors when the application runs. This increased efficiency provides for improved maintenance over the life of the solution. In addition, AppendPDF Pro 3.0 also offers the option of supporting linearization of PDF, sometimes called optimization, so a user can view the first page of a PDF document while the document is still being down-

loaded from the Internet. If a user decides the information needed is on page 25, the user can quickly link to it while the download process continues in the background. Now AppendPDF Pro supports 40- and 128-bit encryption as well as setting user permissions on PDF documents.

www.appligent.com

Vordel and Chrysalis-ITS Collaborate on Integrated XML Security Appliance

(London) – Vordel, the XML security company, and Chrysalis-ITS, the leading provider of hardware security modules, servers, and appliances, announced that they have been working jointly to produce a fully integrated hardware XML security appliance. This announcement is the first project of a partnership agreement between the two companies.

The appliance, slated for release in the third quarter of



this year, will use the powerful cryptographic processing, secure key management, and the market-leading access control and anti-tamper protection of Chrysalis-ITS' Ultimate Trust Security Platform (UTSP) hardware, which also powers their leading network-attached Luna SA HSM server. The new XML security appliance will safeguard and manage all XML and Web services communications

inside and outside the enterprise by employing scalable, feature-rich, and standards-compliant VordelSecure XML security server software. The integrated offering provides centralized security policy management that communicates to distributed security enforcement points at multiple locations throughout the enterprise.

www.vordel.com

www.chrysalis-its.com

Software AG's Natural Mainframe Development Platform Supports XML

(Reston, VA) – Software AG, Inc., a pioneer in XML solutions, announced that version 4 of its Natural 4GL development platform for mainframes now sup-



ports the creation and processing of Extensible Markup Language (XML) documents. Support for XML in Natural Version 4 allows developers to bring mainframe applications into the future by extending them directly to the Internet, the Web, and Web services.

In addition to supporting XML, Natural Version 4 provides several other important new capabilities, including improved overall application performance, accelerated and simplified access to Software AG's Adabas by other databases, the ability to develop applications for a Unix production environment using a Windows desktop, and expanded interoperability between

Natural for IBM's OS/390 or z/OS and Natural for Linux.

www.softwareag.com

XMLMentor to Provide an OTA Solution for Open Enterprises

(Montreal) – XMLMentor, a leading provider of XML education and consulting services, announced that it will offer a



comprehensive training and assessment solution for open enterprises.

Companies around the globe are embracing open-source software and XML standards-based solutions at a fast pace. XMLMentor aims to help these companies in training and certifying their IT professionals on open-source software and XML standards.

www.xmlmentor.net

W3C News

XHTML 2.0 Working Draft Published

The HTML Working Group has released the fifth public working draft of XHTML 2.0. A modularized language without



presentation elements, XHTML 2 takes HTML back to its roots in document structuring. This draft includes an early implementation of XHTML 2.0 in RELAX NG. Comments are welcome.

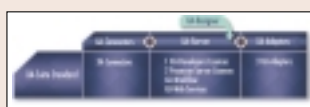
XML Key Management Requirements Published

The XKMS Working Group has published their final set of XML Key Management (XKMS 2.0) Requirements as a W3C Note. The Note documents the design, scope, and requirements of the XML Key Management specification. XKMS specifies protocols for distributing and registering public keys for use with XML Signature and XML Encryption.

www.w3.org

XAware Announces Next-Generation Enterprise Information Integration Server

(Colorado Springs, CO) – XAware, Inc., a leader in XML-based information exchange and data integration, has announced XA-Suite Release 3.0 for Java with advanced Enterprise Information Integration (EII) capabilities. Release 3.0 includes advanced features for data transformation, information security, and simplified access to enterprise



applications such as Siebel and SAP.

"Using XA-Suite, we quickly developed new information exchange processes that integrated data from different systems in a common XML Schema – all in a matter of

hours not days," said Gene Thibodeau, national project manager of Office Technology Systems (OTS), a government sector systems integrator based in Denver. "XAware technology is very compelling when we considered the manpower involved to accomplish the same results in a traditional manner."

www.xaware.com

Quest Software, Inc.

<http://java.quest.com/jcsc/ws>

Altova

<http://xmlj.altova.com/authentic5>