

I Seminarium PLOUG
Warszawa
Marzec 2001

Czy już warto używać XML?

dr inż. Tomasz Traczyk
ttraczyk@ia.pw.edu.pl

Instytut Automatyki i Informatyki Stosowanej
Politechnika Warszawska

Autor

Jest adiunktem Instytutu Automatyki i Informatyki Stosowanej Politechniki Warszawskiej.

Streszczenie

Odpowiadając na tytułowe pytanie, tutorial prezentuje obecny stan rozwoju języka XML oraz narzędzi z nim związanych. Pokazane zostaną wzięte z życia przykłady zastosowań XML, wraz z praktycznymi sposobami użycia odpowiednich narzędzi.

Wprowadzenie

Jedną z najlepiej przyjętych koncepcji informatycznych, które pojawiły się w ostatnich latach, jest język XML. Szybką akceptację zawdzięcza XML z jednej strony prostocie i pewnemu podobieństwu do powszechnie znanego HTML, z drugiej strony faktowi, iż stanowi on niezłe rozwiązanie wielu aktualnych problemów, związanych przede wszystkim z wymianą danych w sieci Web.

Wydaje się, że pozycja języka XML jest już ugruntowana, podstawowe specyfikacje tak że już są gotowe albo są na ukończeniu. Pojawiło się też sporo narzędzi umożliwiających użycie XML w systemach informacyjnych, także w systemach z bazami danych. W rozwój tych narzędzi inwestują najwięksi producenci oprogramowania.

Tytułowe pytanie: *czy już warto używać XML* staje się więc pytaniem istotnym.

Podstawy XML

XML (*eXtensible Markup Language*) jest metajęzykiem do definiowania specjalizowanych języków znakowania. Języki te służą zaś do reprezentowania złożonych dokumentów i struktur danych w postaci plików tekstowych, w których strukturę oznaczono specjalnymi znacznikami. Przetwarzanie takich plików – ze względu na to, że są to właśnie pliki tekstowe oraz że XML zaprojektowano w odpowiedni sposób – jest stosunkowo łatwe i mo żliwe do wykonania za pomocą standardowych i dostępnych narzędzi.

Czym jest XML

W XML można definiować języki opisu stron i/lub języki służące do zapisu danych wraz ze strukturą. XML jest nieco zmodyfikowanym podzbiorem SGML, stąd jego podobieństwo do języka HTML, jedynie zresztą powierzchowne, gdyż cele tych języków są odmienne.

Język XML ma umożliwiać łatwe i precyzyjne definiowanie składni specjalizowanych języków do zapisu różnego rodzaju dokumentów i struktur danych, wyświetlanie takich dokumentów przez przeglądarki, zwłaszcza w sieci WWW, oraz użycie różnego typu powiązań między dokumentami.

Najważniejszym, choć nie jedynym, przeznaczeniem XML jest prezentowanie dokumentów i danych w WWW. Ponieważ te same dane mogą wymagać – w zależności od środowiska prezentacji i od potrzeb odbiorcy – różnych sposobów przedstawienia, znakowanie w XML całkowicie oddzielono od definiowania sposobu prezentacji. Prezentację określa się za pomocą języka XSL „stowarzyszonego” z XML. XSL (*eXtensible Stylesheet Language*) nadaje się także świetnie do przetwarzania dokumentów w XML, np. zamiany ich na HTML.

Dokument w XML

Dokument w XML jest plikiem tekstowym, składa się z prologu, w którym określa się m.in. wersję języka XML i typ dokumentu, oraz z zawartości mieszczącej element główny, w którym zagnieżdżone są pozostałe elementy dokumentu.

Elementy wyróżnione są znacznikami (*tags*). Każdy element może mieć parametry zwane atrybutami. Zawartość elementu stanowi tekst mogący zawierać znaczniki.

Strukturę dokumentu definiuje się za pomocą DTD (*Document Type Definition*) albo tzw. schematów.

DTD i schematy

DTD (*Document Type Definition*) zawiera definicje wszystkich elementów używanych w dokumencie. Z DTD program interpretujący dokument otrzymuje informację o prawidłowej składni dokumentu, tj. o nazwach elementów, ich następstwie i sposobie zagnieżdżenia, atrybutach elementów itp.

W XML istnienie DTD nie jest obowiązkowe, zakłada się bowiem, że przeglądarki powinny umieć odczytać i wyświetlić poprawnie zbudowany dokument nawet jeśli nie mają dostępu do jego DTD.

DTD jest pomysłem dość starym, wywodzi się bowiem wprost z SGML. Nie przewidziano w nim w ogóle możliwości określania dokładnych typów danych – wszystkie elementarne „komórki” danych są po prostu napisami. Tymczasem jeśli w XML reprezentowane są złożone dane, to powinna istnieć możliwość precyzyjnego zdefiniowania typów. Taką możliwość stwarzają tzw. schematy. Definiują one składnię dokumentu wraz z typami poszczególnych „komórek”. Taka definicja ma postać zupełnie różną od DTD, gdyż jest także napisana w XML.

Choć wszystko na to wskazuje, że schematy w przyszłości wyprą DTD, na razie ich specyfikacja (*XML Schema* [11]) nie jest ustabilizowana, a w większości narzędzi nie ma dla nich wsparcia.

Encje

W dokumentach XML można używać tzw. encji (*entities*), które stanowią formę makroinstrukcji, pozwalając definiować stałe fragmenty tekstu lub odwołania do zewnętrznych źródeł w celu ich późniejszego – najczęściej wielokrotnego – użycia. Encje definiuje się w DTD, definicja może określać bezpośrednio rozwinięcie encji lub adres pliku zawierającego owo rozwinięcie. Encji używa się często – podobnie jak w HTML – do definiowania znaków specjalnych (dostępne są nieliczne encje predefiniowane, np. `<`; `"`;). Odwołanie do encji poprzedzone jest znakiem `&`, kończy się zaś średnikiem.

Poprawność dokumentu w XML

Dokument w XML może osiągnąć dwa stopnie poprawności. Pierwszy z nich oznacza, że dokument jest na tyle kompletny, iż może być zinterpretowany przez przeglądarkę. Drugi oznacza pełną poprawność składniową struktury dokumentu.

Dokument poprawny w pierwszym sensie nazywa się dobrze sformułowanim (*well-formed*). Musi on spełniać warunki określone dla tego stopnia poprawności w specyfikacji XML. Najważniejsze z nich to nakaz jawnego zakończenia wszystkich rozpoczętych konstrukcji językowych i oznaczania znaczników pustych oraz zakaz „krzyżowania” elementów (każda konstrukcja zawarta w innej, musi być w niej zawarta w całości). Dokument taki nie musi natomiast być zgodny z DTD, ani nawet posiadać DTD. Dokumenty z założenia nie wyposażone w DTD powinny być poprawne w tym sensie.

Dokumenty dobrze sformułowane mogą być zinterpretowane przez analizator leksykalny (*parser*) i wyświetlone przez przeglądarkę bez konieczności dostępu do DTD.

Dokument przeznaczony do przetwarzania powinien być poprawny w drugim sensie. Taki dokument nazywa się prawidłowym (*valid*). Prawidłowość dokumentu jest uwarunkowana istnieniem DTD powiązanego z dokumentem, pełną zgodnością zawartości dokumentu z DTD oraz poprawnością w sensie *valid* wszystkich konstrukcji użytych w dokumencie, jak to określono w szczegółowej specyfikacji języka. Oczywiście dokument prawidłowy musi być także dobrze sformułowany.

Poprawność w sensie *valid* sprawdzają tzw. parsery walidujące. Programy przetwarzające dane z dokumentu – zwykle budowane w oparciu o parsery XML – na ogół powinny wymagać tego stopnia poprawności.

Przestrzenie nazw

Choć od powstania XML minęło niewiele czasu, stworzono już bardzo wiele systemów (słowników) znaczników, przeznaczonych do stosowania w różnych specyficznych dziedzinach. Autor budujący dokument z danej dziedziny powinien móc korzystać z już istniejących systemów znaczników, ale może potrzebować znaczników z więcej niż jednego słownika, może także chcieć dołączyć swoje własne znakowanie. W takiej typowej sytuacji może dojść do konfliktów nazw.

Aby zapobiec tego typu problemom, określono sposób wyznaczania i wykorzystania tzw. przestrzeni nazw (*XML namespaces*) [10].

Przestrzeń nazw jest jednoznacznie identyfikowana przez podanie URI (*Uniform Resource Identifier*, czyli adresu sieciowego) domeny, która zarządza daną przestrzenią nazw.

Element XML odwołuje się do przestrzeni nazw przez podanie specjalnego atrybutu `xmlns` i zdefiniowanie prefiksu, który będzie służył do wyróżniania znaczników należących do danej przestrzeni nazw. W jednym dokumencie można powołać się na wiele przestrzeni nazw.

Przetwarzanie dokumentów w XML

Dokumenty zapisywane są w XML po to, by mogły być efektywnie przetwarzane przez standardowe oprogramowanie. Najbardziej powszechną formą przetwarzania jest niewątpliwie prezentacja dokumentu w przeglądarce. Inne typowe przetwarzanie to wykorzystanie danych z dokumentu w programach. Opracowano więc odpowiadające tym typom przetwarzania standardy, którym powinno podlegać oprogramowanie przeznaczone do współpracy z XML.

Style-sheets i XSL

Dokument w XML powinien być zbudowany na zasadzie znakowania znaczeniowego, a nie typograficznego. Cała informacja o sposobie formatowania dokumentu przez przeglądarkę musi być zatem sformułowana osobno.

Do określenia wyglądu dokumentów służą tzw. arkusze stylistyczne (*style-sheets*). Określają one sposób prezentacji każdego z elementów. Do definiowania arkuszy stylistycznych stworzono język XSL (*eXtensible Stylesheet Language*). Możliwe jest także definiowanie prezentacji dokumentów w XML za pomocą języka CSS.

XSL (*eXtensible Stylesheet Language*) [8] jest językiem służącym do prezentacji dokumentów w XML i do ich przekształcania.

Składnia XSL została zdefiniowana w XML, z użyciem przestrzeni nazw (*namespaces*). XSL składa się z dwóch części: języka transformacji XSLT (*XSL Transformations*) [9] oraz słownika obiektów specyfikujących formatowanie dokumentu (*formatting objects*).

Istotę działania XSL stanowi rekurencyjne przetwarzanie znaczników. Prezentacja dokumentu z użyciem XSL przebiega w dwóch fazach: transformacji drzewa znaczników na tzw. drzewo wynikowe i przypisania elementom tego drzewa sposobu prezentacji. Znaczniki XML wejściowego dokumentu są w tym procesie identyfikowane za pomocą wzorców (*templates*), które mogą w elastyczny sposób określać miejsce znacznika w hierarchii oraz atrybuty znacznika.

Druga faza procesu przetwarzania nie jest obowiązkowa, XSL może więc być z powodzeniem używany do przekształcania dokumentów w XML. Ponieważ przeglądarki, za pomocą których prezentowane są dokumenty XML, są zapewne przystosowane do prezentacji HTML, wygodnym sposobem formatowania dokumentów za pomocą XSL jest przetworzenie ich na HTML.

DOM i SAX

Dane zawarte w dokumentach XML powinny dać się wygodnie przetwarzać w programach – potrzebny jest do tego standard dostępu do danych i uniwersalne narzędzia ułatwiające ten dostęp.

Propozycją takiego standardu jest DOM (*Document Object Model*). Definiuje on obiektowy model dokumentu w XML oraz dostarcza zbioru klas i metod umożliwiających manipulowanie dokumentami XML z poziomu języków programowania.

W DOM dokument XML jest reprezentowany jako drzewo obiektów, na którym można wykonywać różnorodne operacje. Wadą tego podejścia jest wielkość zasobów pamięci wymaganych do reprezentowania dużych dokumentów, zaletą – możliwość programowej modyfikacji przetwarzanych dokumentów.

API zgodne z DOM (np. wbudowane w przeglądarki WWW) umożliwiają wygodne manipulowanie dokumentami w typowych środowiskach programowania. Najbardziej znaną implementacją DOM jest część MSIE 5.

Popularność zdobył także, spełniający podobne jak DOM zadanie, model SAX (*Simple API for XML*). SAX opiera się na programowaniu zdarzeniowym: zdarzenia wywoływane są np. przez pojawienie się początku i końca elementu w analizowanym dokumencie. Nie jest więc potrzebne tworzenie struktury danych reprezentującej cały dokument, co powoduje małe zużycie zasobów i znaczną wydajność. Wadą jest jednak niemożność modyfikowania przetwarzanych dokumentów za pomocą SAX.

Standaryzacja i rozwój języka XML

Standaryzacją języka XML i języków mu towarzyszących (np. XSL) zajmuje się organizacja *World Wide Web Consortium* (W3C). Zatwierdza ona zgłoszone propozycje standardów. Do organizacji tej zgłaszane są także wszelkie propozycje rozszerzeń języka.

Obecnie specyfikacje języka XML, przestrzeni nazw, XSLT oraz modelu DOM mają status *W3C Recommendation*, praktycznie równoważny z uznaniem standardu. Mniej zaawansowane są prace nad specyfikacjami schematów – mają status *W3C Candidate Recommendation*.

XML przeżywa wciąż szybki rozwój. Specyfikacje podlegają ewolucji, a na ustabilizowanie się kompletu standardów przyjdzie zapewne jeszcze poczekać. Pojawienie się na rynku powszechnie dostępnych narzędzi tworzy jednak pewne standardy *de facto*. Na szczęście producenci narzędzi na ogół starają się dostosowywać swoje wyroby do ustaleń W3C, jest więc nadzieja, że nie powstaną – jak w wielu innych dziedzinach informatyki – wiele niezgodnych ze sobą rozwiązań firmowych.

Narzędzia dla XML

Język jest wówczas przydatny praktycznie, gdy istnieją dostępne i sprawdzone narzędzia umożliwiające jego wykorzystanie. W przypadku języka XML można już stwierdzić, że obecnie podstawowe narzędzia istnieją i są dostępne; co więcej, są one dostarczane przez największych producentów oprogramowania.

Podstawowe narzędzia dla XML to:

- parsery XML dla języków programowania, np. Java, C++;
- procesory XSLT;
- edytory do plików XML, ułatwiające manipulowanie strukturą danych;
- przeglądarki, umożliwiające bezpośrednie wyświetlanie dokumentów w XML.

Istnieje wiele parserów XML i procesorów XSLT oraz edytorów do XML. Część z nich to produkty typu *open source*, część jest komercyjna. W tym artykule skupimy się na produktach komercyjnych dostarczanych przez Microsoft (bezpłatnie!) oraz przez Oracle.

W przypadku przeglądarek sytuacja jest zdecydowanie gorsza. W miarę kompletną obsługę XML i XSLT zawiera jedynie *Microsoft Internet Explorer*. W żadnej z popularnych przeglądarek nie zaimplementowano obsługi obiektów formatujących XSL.

MSIE 5

Najbardziej znanym produktem związanym z XML jest *Microsoft Internet Explorer 5* (MSIE). MSIE potrafi prezentować dokumenty XML z wykorzystaniem arkuszy stylistycznych napisanych w językach XSL oraz CSS. Zrealizowano jedynie pierwszą część przetwarzania XSL – transformację, nie zaimplementowano formatowania za pomocą obiektów formatujących.

MSIE potrafi także w dogodny sposób wyświetlać dokumenty XML bez zdefiniowanych arkuszy stylistycznych; dokumenty takie są prezentowane w formie rozwijalnej hierarchii.

W podstawową wersję MSIE wbudowano procesor XSL działający według starszej wersji specyfikacji XSL, bez ważnych rozszerzeń wprowadzonych w specyfikacji XSLT. Walidacja dokumentów jest wprawdzie możliwa, ale jedynie programowo (za pomocą skryptu w języku JScript), nie można jej wywołać wprost z przeglądarki. Wyniki transformacji z XML na HTML są prezentowane w przeglądarce, ale nie można obejrzeć otrzymanego kodu HTML.

Na szczęście możliwe jest usunięcie tych wad przez zainstalowanie odpowiednich uzupełnień dostarczanych przez producenta, a dostępnych w jego serwisie WWW [12].

Instalacja uzupełnień MSIE

W celu przystosowania MSIE do wygodnego przetwarzania i prezentacji dokumentów XML należy zainstalować następujące poprawki.

- *Internet Explorer Tools for Validating XML and Viewing XSLT Output* – uzupełnia MSIE o opcje (dostępne w menu wywoływanym prawym przyciskiem myszy), umożliwiające walidację prezentowanego dokumentu XML oraz wyświetlenie w osobnym oknie kodu (zwykle w HTML) otrzymanego w wyniku transformacji XSLT.
- *MSXML Parser 3.0* – jest uaktualnieniem parsera XML oraz procesora XSL; zawiera API DOM i SAX 2 oraz procesor obsługujący zarówno starszą wersję XSL (tę samą, której obsługę wbudowano oryginalnie w MSIE), jak i wersję zgodną ze specyfikacją XSLT.
- *xmllnst.exe Replace Mode Tool* – jest to narzędzie umożliwiające przekonfigurowanie MSIE tak, by poprawnie przetwarzał on arkusze stylistyczne zgodne ze specyfikacją XSLT. Sama instalacja parsera 3.0 bez zastosowania tego narzędzia umożliwia wykorzystanie XSLT jedynie w skryptach, natomiast po przekonfigurowaniu MSIE potrafi automatycznie (na podstawie instrukcji przetwarzania `xml-styleSheet`) wykorzystywać zarówno arkusze w starszej wersji XSL jak i w XSLT. Rozróżnienie wersji następuje wg adresu URI podawanego w definicji przestrzeni nazw w arkuszu stylistycznym.

MSIE z zainstalowanymi uzupełnieniami, wraz z opisanymi niżej dodatkowymi narzędziami, tworzy środowisko dostarczające podstawowych środków do pracy z XML.

Narzędzia pomocnicze

Z MSIE związane są przydatne narzędzia pomocnicze.

- *MSXML SDK 3.0* – pakiet zawierający parser, pliki nagłówkowe do API, dokumentację do XML, XSL i SAX2 oraz przykłady w językach JScript, Visual Basic i C++.
- *MSXSL Command Line Transformation Utility* – program umożliwiający wywołanie procesora XSLT z linii poleceń i dokonywanie transformacji plików XML za pomocą skryptów XSLT (np. transformacji XML na HTML).

- *XML Validation Tool* – program umożliwiający przeprowadzenie walidacji pliku XML, wywoływany z linii poleceń.
- *XSL to XSLT converter* – narzędzie przeprowadzające konwersję skryptów napisanych w starszej wersji XML (uwzględnionej w MSIE 5.0) na XSLT. Służy do uaktualniania przygotowanych wcześniej arkuszy stylistycznych do postaci zgodnej z aktualnym standardem XSLT i możliwościami parsera w wersji 3.0.

XML Notepad

Uzupełnienie MSIE stanowi prosty edytor do plików XML o nazwie *XML Notepad*. Umożliwia on prezentację dokumentu w postaci rozwijalnego drzewa, edycję i łatwe dodawanie elementów i parametrów, kopiowanie gałęzi, dostosowane do składni XML wyszukiwanie i zamianę tekstu. Dokumenty są przez edytor automatycznie walidowane.

Oracle XDK

Korporacja Oracle szybko zauważyła znaczenie języka XML. Produkty Oracle zostały więc w ostatnich latach wyposażone w elementy umożliwiające wykorzystanie XML [13].

Podstawowy zestaw narzędzi wspomagających wykorzystanie XML nazwany został *Oracle XML Developer's Kit* (XDK). W skład tego pakietu wchodzi:

- parsery XML dla różnych języków programowania;
- procesor XSLT;
- procesor schematów *XML Schema*;
- generatory klas dla języków obiektowych;
- gotowe rozwiązania: narzędzie *XML SQL Utility*, pakiet komponentów *Java Beans* i serwet realizujący zapytania SQL.

XDK jest dostarczany w czterech wersjach, dla języków programowania Java, C++, C i PL/SQL. Wersje te różnią się funkcjonalnością, dlatego opisane zostały osobno.

Oracle XDK for Java

Najpełniejszy pakiet dostarczono dla języka Java. Zawiera on następujące produkty.

- *XML Parser/XSLT Processor for Java v. 2* – wydajny analizator leksykalny zgodny ze standardami DOM Level 1, SAX 1.0, XSLT [9] oraz *XML Namespaces* [10]. Pracuje w trybie z walidacją i bez niej. Parser ten może być wykorzystywany przez programy w Javie działające we wszystkich warstwach architektury trójwarstwowej, także w składowanych w bazie procedurach w Javie. Działa z wieloma stronami kodowymi, tak że ze stroną ISO-8859-2, zawierającą polskie znaki diakrytyczne.
- *XML Parser for Java v. 2.0.2.9 beta* – nowa wersja parsera, dodatkowo zawierająca DOM2 (częściowo) oraz SAX 2.0.
- *Oracle XML Schema Processor for Java* – uzupełnienie parsera, pozwalające analizować typy danych w dokumencie XML, zgodnie ze specyfikacją *XML Schema* [11].
- *XML Class Generator for Java* – narzędzie budujące na podstawie DTD zestaw klas języka Java, służących to tworzenia, walidacji i drukowania zgodnych z podanym DTD dokumentów XML¹. W wersji 2 zamiast DTD podany być może schemat *XML Schema*.
- *Oracle XML Transviewer Beans* – zestaw komponentów umożliwiających budowanie graficznych interfejsów do aplikacji XML; więcej szczegółów podano dalej.

¹ Obecna wersja nie działa ze stroną kodową ISO-8859-2.

- *XML SQL Utility for Java (XSU)* – zestaw gotowych klas służących do generowania dokumentów w XML na podstawie zapytań w SQL oraz do wczytywania dokumentów w XML do struktur bazy danych; więcej informacji podano niżej.
- *XSQL Servlet* – gotowa aplikacja, oparta na XSU, przeznaczona do wykonania na serwerze danych lub aplikacji (*servlet*). Umożliwia ona wstawianie zapytań SQL wprost do dokumentów w XML i dynamiczne wykonywanie tych zapytań; pozwala ona także na dokonywanie przez serwer transformacji dokumentów w XML na podstawie skryptów w XSLT.

Oracle XDK for C

Dla języka C dostarczone są następujące narzędzia.

- *XML Parser/XSLT Processor for C v. 2* – o właściwościach podobnych do parsera dla Javy, zgodny z DOM Level 1 i SAX 1.0.
- *XML Schema Processor for C* – uzupełnienie parsera dla C, zgodne ze specyfikacją *XML Schema* [11].

Oracle XDK for C++

Dla C++ Oracle dostarcza narzędzia wymienione niżej.

- *XML Parser/XSLT Processor for C++ v. 2* – o właściwościach podobnych do parsera dla Javy, zgodny z DOM Level 1 i SAX 1.0.
- *XML Schema Processor for C++* – uzupełnienie parsera dla C++, zgodne ze specyfikacją *XML Schema* [11].
- *XML Class Generator for C++* – narzędzie budujące na podstawie DTD zestaw klas języka C++, służących to tworzenia, walidacji i drukowania zgodnych z podanym DTD dokumentów².

Oracle XDK for PL/SQL

Dostępny jest także parser XML dla języka PL/SQL. Parser ten jest napisany częściowo w języku Java, działa więc w oparciu o Oracle8i Java VM. Parser zawiera API DOM Level 1 oraz procesor XSLT. Niestety, nie ma możliwości wykorzystywania przestrzeni nazw. Nie ma także przetwarzania w modelu SAX, co wynika jednak z ograniczeń samego języka PL/SQL – SAX wymaga bowiem programowania zdarzeniowego.

Oracle XML Transviewer Beans

Produkt ten jest zestawem gotowych komponentów typu *Java Beans*, służących do tworzenia interfejsu graficznego do aplikacji XML.

W skład produktu wchodzi:

- *DOMBuilder Bean* – interfejs do parsera DOM, zapewniający asynchroniczną analizę leksykalną;
- *TreeViewer Bean* – komponent wyświetlający struktury XML w formie drzewa, z możliwościami manipulacji za pomocą myszki;
- *SourceViewer Bean* – komponent wyświetlający źródłowe pliki w XML, z wyróżnianiem elementów składni za pomocą kolorów;
- *Transformer Bean* – komponent dokonujący transformacji XML na podstawie podanego skryptu XSL.

Oracle XML SQL Utility (XSU)

Jest to zestaw klas języka Java realizujących następujące funkcje.

² Obecna wersja nie działa ze stroną kodową ISO-8859-2.

- Generowanie dokumentów XML (w postaci tekstu lub drzewa DOM) na podstawie zapytania SQL albo obiektu *ResultSet* JDBC. Dokument jest generowany w tzw. postaci kanonicznej.
- Wyodrębnianie danych z dokumentów XML i użycie ich do wstawienia, modyfikacji lub kasowania wierszy w tabeli bazy danych.

Narzędzie zapewnia także:

- obsługę wszystkich typów danych występujących w tabelach bazy danych Oracle;
- dynamiczne generowanie DTD i schematów *XML Schema*;
- proste przekształcenia w czasie przetwarzania, np. zmianę nazwy znacznika;
- możliwość współpracy z procesorem XSLT;
- możliwość wygenerowania ciągu wywołań SAX, reprezentującego dokument XML;
- generowanie atrybutów XML – wybrane dane z bazy danych mogą być przekształcane w atrybuty, a nie elementy.

Do działania narzędzia niezbędny jest parser XML dla języka Java v. 2 oraz działający interfejs JDBC, zapewniający łączność z bazą danych.

Narzędzie może być używane w środowisku języka Java w każdej z warstw systemu informacyjnego: w bazie danych, na serwerze aplikacyjnym, jako servlet serwera Web lub na kliencie. Posiada także interfejs do języka PL/SQL, może więc działać w bazie danych, wywoływane z tradycyjnych aplikacji PL/SQL. Narzędzie posiada także interfejs pozwalający na użycie go bezpośrednio z linii poleceń systemu operacyjnego.

XML w innych produktach Oracle

Oracle DBMS

W DBMS Oracle8i wbudowano maszynę wirtualną języka Java (Oracle8i Java VM). Dzięki temu możliwe jest składowanie i wykonywanie w bazie danych programów wykorzystujących XDK.

Od wersji 8.1.7 bazy danych Oracle8i narzędzia dla XML stały się częścią pakietu systemu zarządzania bazą danych (DBMS) i są dostarczane razem z nim.

Specjalną obsługę XML wbudowano także do opcji *interMedia Text*: dostosowano ją do wyszukiwania w dokumentach w XML, np. dodano specjalny operator *nest within*.

Oracle Internet File System (iFS)

Specjalne rozwiązania dla XML wbudowano także w opcję *iFS*. Po odpowiednim skonfigurowaniu *iFS* może automatycznie analizować zawartość ładowanych dokumentów XML i zapisywać je wprost do tabel bazy danych w postaci ustrukturalizowanej. Gdy użytkownik chce pobrać taki dokument z *iFS*, następuje automatyczne jego odtworzenie (*rendering*).

Oracle JDeveloper 3.1

Programiści tworzący aplikacje za pomocą pakietu JDeveloper mogą korzystać z narzędzi zawartych w *Oracle XDK for Java*, z komponentów *Oracle XML Transviewer Beans* i z modułu *XSQL Servlet*.

Stosowany w JDeveloper model komponentowy BC4J używa wewnętrznie języka XML do przechowywania metadanych, takich jak reguły przetwarzania (*business rules*). Dostarczany jest także komponent *Web Bean*, który może być używany w servletach i aplikacjach typu JSP. Komponent ten czyta dane z komponentów typu BC4J i przekształca je na XML, może także używać arkusza XSLT do transformacji wyjściowego dokumentu.

Oracle Reports 6i

Od wersji 6i narzędzie Oracle Reports jest wyposażone w możliwość wyprowadzania raportów w języku XML.

Zastosowania XML

W ciągu kilku lat istnienia XML stał się popularny i znalazł liczne zastosowania. W tym artykule przedstawione zostaną podstawowe typy zastosowań, wraz z wziętymi z doświadczenia autora przykładami.

Specjalizowane struktury danych

XML wydaje się idealnym środkiem do formułowania i wymiany specjalistycznych dokumentów i struktur danych. Powstaje więc wiele języków specjalizowanych opartych na XML. Dotyczą one wielu bardzo różnych dziedzin. Kilka przykładów podano poniżej:

- zastosowania naukowe, np. MathML (*Mathematical Markup Language*), CML (*Chemical Markup Language*);
- modelowanie systemów, np. PIF-XML (Process Interchange Format XML), UXF (UML eXchange Format), XMI (XML Metadata Interchange);
- finanse i bankowość, np. OFX/OFE (*Open Financial Exchange*), BIPS (*Bank Internet Payment System*);
- multimedia, np. SVG (*Scalable Vector Graphics*), SMIL (*Synchronized Multimedia Integration Language*), PGML (*Precision Graphics Markup Language*).

XML w elektronicznej wymianie danych

Ważnym polem zastosowania XML jest elektroniczna wymiana danych. Spotkać można trzy podejścia do tego zagadnienia:

- Projektowanie własnych struktur, specyficznych dla danego zagadnienia. To podejście jest stosunkowo łatwe do zastosowania i daje duże możliwości specjalizacji struktur; jednak ze względu na brak oparcia na standardach może utrudniać wymianę informacji w szerszych społecznościach.
- Zastosowanie jednego z powstających standardów XML-owych. Podejście to można zastosować, jeśli istnieje jakaś propozycja standardu o potrzebnych właściwościach. Wymiana danych w szerszych społecznościach jest tu potencjalnie łatwiejsza, ale większość propozycji standardów XML-owych nie jest jeszcze stabilna i – poza nielicznymi wyjątkami – nie są one rozpowszechnione.
- XML-EDI – w tym podejściu semantyka dokumentów jest wzorowana na komunikatach EDI (*Electronic Data Interchange*), np. wg standardu EDIFACT, zapisuje się je zaś w składni XML. To podejście pozwala wykorzystać doświadczenia i osiągnięcia EDI, jest jednak dość trudne.

Zastosowanie XML w elektronicznej wymianie danych ma liczne zalety, m.in.:

- Komunikaty mogą być przetwarzane standardowymi narzędziami, z których większość jest bezpłatna; pozwala to na zastosowanie tej technologii nawet przez małe firmy – inaczej niż w przypadku typowych rozwiązań EDI, które z reguły były bardzo kosztowne.
- Ze względu na tekstową postać i istnienie DTD łatwo jest ocenić formalną poprawność (np. kompletność) przekazanych danych.
- Komunikaty są czytelne i na ogół mogą być zinterpretowane nawet bez dostępu do szczegółowego opisu.

- Istniejące propozycje standardów mogą być dość łatwo rozbudowywane; szczególnie przydatne jest tu zastosowanie przestrzeni nazw.
- Komunikaty zapisane w nowszych wersjach standardów mogą być interpretowane przez starsze oprogramowanie, jeśli tylko przy projektowaniu nowszych wersji nie zmieniono znaczenia poprzednio istniejących znaczników. Narzędzia przetwarzające na ogół bowiem stosują znaną z HTML zasadę, iż elementy nieznanne są ignorowane.
- XML jest uznanym standardem, w dodatku niezbyt skomplikowanym – nie będzie więc trudno o specjalistów umiejących się posługiwać tą technologią.

Podstawową wadę XML w zastosowaniu do elektronicznej wymiany danych stanowi duży rozmiar dokumentów – zawierają one bowiem znaczny „narusz” spowodowany przez znaczniki. Jednak wobec możliwości kompresji danych na czas przesyłu ta wada nie wydaje się zbyt znacząca. Innym problemem jest ograniczona wydajność narzędzi do przetwarzania XML, jednak jest to zapewne zjawisko przejściowe, gdyż w technologii XML-owe inwestują najwięksi producenci oprogramowania.

Przykład 1

Postanowiono użyć języka XML do gromadzenia danych dotyczących opieki zdrowotnej, np. danych o sprzedaży leków refundowanych, o udzielonych usługach medycznych itp. Opracowano (przy współudziale autora) pierwsze propozycje komunikatów, które znalazły odzwierciedlenie w odpowiednim rozporządzeniu Ministra Zdrowia [6].

XML w analizie systemów informacyjnych

Specyficzny rodzaj zastosowania może znaleźć XML w analizie systemów informacyjnych. Otóż we wczesnych fazach analizy zwykle tworzy się modele bazujące na tekście, w formie opisów, tabel, zestawień itp. W dodatku na tym etapie często nie jest jeszcze podjęta decyzja o zastosowaniu konkretnych narzędzi wspomagających projektowanie, np. narzędzi CASE.

Analitik powinien jednak w miarę możliwości tworzyć modele sformalizowane. Potrzebuje narzędzia do tworzenia zestawień, tabel, różnych przekrojów informacji, a także do produkcji profesjonalnie wyglądających raportów – zarówno drukowanych, jak prezentowanych przez WWW. Do tego celu doskonale nadaje się właśnie XML i standardowe – dostępne za darmo – narzędzia do jego przetwarzania.

Tworząc modele analitik może zbudować swoje własne struktury danych w XML, odzwierciedlające potrzebną formalizację. Elementy opisowe mogą być odpowiednio oznakowane, by możliwe było ich czytelne sformatowanie (oczywiście powinno się stosować znakowanie znaczeniowe, a nie typograficzne).

Struktur dokumentów XML zwykle nie projektuje się od początku dla każdego projektu. Możliwe jest łatwe użycie „uniwersalnych” struktur, użytecznych przy różnych projektach. Często także specjalistyczne struktury użyte w poprzednich projektach dadzą się – po niewielkich zmianach – przystosować do nowych potrzeb.

Po zaprojektowaniu struktur XML należy stworzyć arkusze stylistyczne w XSLT, przekształcające wejściowe dane w przydatną postać wynikową. Zwykle tworzy się kilka takich arkuszy, prezentujących informację w różnych formach oraz przystosowanych do różnych sposobów prezentacji:

- prezentacji w WWW (transformacja XML w HTML);
- tworzenia roboczych raportów (transformacja XML w HTML, a następnie w RTF);
- druku profesjonalnej dokumentacji (transformacja XML np. do dokumentów LaTeX-owych).

Typowe powtarzalne elementy analizy, w których można z powodzeniem użyć XML to np.:

- tworzenie słowników pojęć, skrótów, osób, instytucji;
- rejestrowanie spotkań (wywiadów), np. tzw. *minutes*.

Model w XML przydaje się także do planowania samego procesu analizy, gdy trzeba zwykle:

- przeprowadzić analizę wymagań na podstawie umowy (często zawierającej bardzo wiele szczegółowych postulatów);
- stworzyć plan działań i podział zadań;
- zaprojektować raport (lub raporty, jeśli analiza ma wiele etapów);
- sprawdzić wzajemną zgodność wymagań, planu działań i projektu raportu.

Typowe programy do planowania przedsięwzięć nie zawierają wsparcia dla tego typu planowania, a przy bardziej skomplikowanych problemach jest ono niezbędne.

Przykład 2

Oto kilka przykładów zastosowania XML w analizie systemów informacyjnych.

1. Analiza procesów biznesowych do celów eksploracji danych (*data mining*).

W XML sformułowano m.in.

- sformalizowany spis procesów;
- spis i ocenę zasobów danych;
- zależności między danymi a procesami;
- ocenę „podatności” procesów na eksplorację danych (skrypt XSL wyliczał oceny zbiorcze na podstawie kryteriów cząstkowych).

2. Analiza wymagań funkcjonalnych i techniczno-organizacyjnych dla internetowych systemów informacyjnych.

XML użyto m.in. do:

- stworzenia sformalizowanego wykazu wymagań wobec systemu;
- analizy rozwiązań wariantowych;
- klasyfikacji źródeł i rodzajów informacji oraz odbiorców informacji.

Używając podobnej metodologii i struktur XML prowadzono następujące projekty:

- projekt krajowego systemu promocji eksportu;
- projekt ogólnopolskiej sieci innowacyjnej z dziedziny nowych technologii;
- projekt portalu krajowego samorządu gospodarczego.

3. Porównanie cech wybranych systemów zarządzania bazami danych ze względu na przydatność do zastosowań w wielkim eksperymencie fizyki jądrowej.

W XML stworzono struktury zawierające:

- wykaz kryteriów z objaśnieniami i dwustopniowym podziałem na kategorie;
- spis branż pod uwagę systemów zarządzania bazami danych;
- listę rodzajów zastosowań;
- opis cech każdego z systemów w kontekście każdego kryterium;
- ocenę każdego z systemów w każdej kategorii kryteriów;
- ocenę ważności poszczególnych kategorii kryteriów dla każdego z zastosowań.

XML w systemach z bazami danych

Rozpowszechnienie się sposobu zapisu informacji właściwego dla XML stanowi poważne wyzwanie dla producentów systemów zarządzania bazami danych i narzędzi do tworzenia systemów informacyjnych z bazami danych.

W systemach z bazami danych XML zastosowany być może przede wszystkim do reprezentacji danych poza bazą oraz do przechowywania złożonych struktur w bazie danych.

Reprezentacja danych poza bazą danych

W tym zastosowaniu atutami XML są m.in.:

- łatwy zapis złożonych danych;
- dobre dostosowanie do wymiany danych w systemach heterogenicznych – dokument XML oprócz informacji zawiera także metainformację;
- możliwość bezpośredniej prezentacji w WWW.

To zastosowanie jest dosyć oczywiste i nie nastręcza większych trudności koncepcyjnych. Do jego realizacji wystarczy dostępność interfejsów umożliwiających przekształcanie informacji zgromadzonej w bazach danych w postaci relacyjnej na język XML i odwrotnie. Pierwsze rozwiązania, np. parsery XML zintegrowane z systemami zarządzania bazami danych, już się pojawiły (np. narzędzia Oracle opisane wyżej).

Przechowywanie złożonych struktur w bazie danych

W tym zastosowaniu XML stosować warto, gdyż:

- jest on wybitnie przydatny dla danych o strukturze zmiennej i/lub złożonej;
- możliwy jest zapis złożonych dokumentów w formie CLOB zamiast BLOB;
- zawartość tych dokumentów może być więc przetwarzana w bazie danych w sposób właściwy dla tekstów, np. przez narzędzia pełnotekstowe;
- dokumenty można przechowywać w bazie danych bez utraty jakiejkolwiek informacji o ich strukturze i wyprowadzać różnorodnie sformatowane, zależnie od potrzeb.

Zastosowania tego typu mogą obejmować reprezentację w bazie danych np. złożonych dokumentów tekstowych, arkuszy kalkulacyjnych itp.

Upowszechnienie tego podejścia do reprezentowania złożonej informacji nastręcza jednak spore trudności techniczne i koncepcyjne. Często natrafia się bowiem na struktury, które trudno jest przekształcić do postaci relacyjnej lub ich reprezentacja w pełni relacyjna jest nieefektywna. Konieczne jest więc znalezienie nowych sposobów przechowywania takiej informacji w bazach danych, z zachowaniem możliwości efektywnego wyszukiwania informacji. Zwykle podejmowane są próby przechowywania dokumentów w formie częściowo ustrukturalizowanej, ale nie zawsze spełniają one wymagania efektywności. Upowszechnienie XML może więc spowodować konieczność znacznych zmian w technologii samych baz danych i w sposobach ich stosowania.

Przykład 3

Na dużym wydziale wyższej uczelni funkcjonuje system informacyjny wspomagający proces dydaktyczny. Jednym z elementów tego systemu jest katalog nauczanych na wydziale przedmiotów; ponieważ wydział jest istotnie duży, oferta jest bardzo szeroka i obejmuje kilkaset tytułów.

Studenci mają możliwość zapoznawania się z ofertą wydziału za pomocą WWW. W wydziałowej bazie danych zgromadzone są niezbędne informacje, w tym tzw. konspekty przedmiotów, opisujące zawartość wykładów, tematykę zajęć laboratoryjnych itp. Ze względu na różnorodność prowadzonych zajęć konspekty te mają niejednorodną budowę: pewne elementy struktury są w nich stałe, inne są po prostu tekstem z typowymi elementami formatowania: tytułami części, akapitami, wypunktowaniami, wyróżnieniami itp.

Zdecydowano się tworzyć konspekty w formie dokumentów XML [1],[2]. Teksty nadsyłane przez wykładowców są przez redaktora opracowywane i umieszczane w bazie danych.

Dokumenty są przechowywane w bazie danych w formie częściowo ustrukturalizowanej: stałe elementy struktury są „rozkładane” do postaci relacyjnej, zaś części o zmiennej strukturze są przechowywane jako teksty ze znacznikami.

Przy wyprowadzaniu konspektów z bazy danych następuje odpowiednia konwersja owych tekstów, np. na HTML.

Rozwiązanie to jest już dość stare i działa z powodzeniem od kilku lat. Powstało gdy specyfikacja XML nie była jeszcze stabilna, format dokumentów trzeba więc było poprawiać, by dostosować go do ostatecznej wersji specyfikacji. Gdy rozwiązanie to wdrażano, nie było jeszcze żadnych narzędzi do tworzenia aplikacji XML w systemach z bazami danych, odpowiednie oprogramowanie, np. uproszczony parser, napisano więc samodzielnie – nie okazało się to jednak zadaniem trudnym.

Wykorzystanie narzędzi Oracle do przetwarzania XML

Dostarczane przez Oracle narzędzia mogą być w stosunkowo prosty sposób użyte do budowy systemów informacyjnych wykorzystujących XML.

Gdy XML jest użyty do reprezentacji informacji poza bazą danych, typowy sposób postępowania może wyglądać tak:

- dane znajdują się w tabelach relacyjnych;
- za pomocą perspektywy obiektowej dane z wielu tabel są łączone w złożoną wielopoziomą strukturę.

W celu wyprowadzenia danych z bazy:

- narzędzie XSU generuje plik XML odpowiadający tej strukturze, w postaci kanonicznej;
- procesor XSLT przekształca postać kanoniczną do pożądanej postaci wyjściowej.

Przy wyprowadzaniu danych zamiast perspektywy obiektowej można użyć zapytania SELECT z podzapytaniem typu CURSOR (*cursor subquery*).

Wprowadzanie danych odbywa się podobnie:

- procesor XSLT przekształca wejściowy plik XML do postaci kanonicznej;
- narzędzie XSU wczytuje go używając jako celu perspektywy obiektowej;
- jeśli perspektywa ta jest niemodyfikowalna, to muszą być stworzone wyzwalacze typu `INSTEAD OF`, które wykonają zapis danych do tabel relacyjnych.

Podsumowanie

Język XML zdobył duże uznanie i okazuje się przydatny w licznych zastosowaniach z wielu różnych dziedzin. Jego specyfikacja jest nadal udoskonalana, dodawane są też nowe istotne komponenty, lecz obecnie istnieje już w miarę stabilny zestaw podstawowych standardów, umożliwiający praktyczne użytkowanie języka.

Język zyskał poparcie znaczących producentów oprogramowania, w tym Oracle. Pojawiło się wiele narzędzi umożliwiających tworzenie aplikacji XML. Narzędzia te są często darmowe albo dodawane bezpłatnie do pakietów oprogramowania, np. systemów zarządzania bazami danych.

XML jest ważnym standardem zapisu i przekazywania informacji, a jego wykorzystanie stanie się zapewne już w najbliższych latach powszechne. Szczególnie ważną rolę będzie zapewne odgrywał XML w rozwoju aplikacji internetowych, w szczególności typu *e-commerce*.

Można więc stwierdzić, iż:

- prace nad XML są na tyle zaawansowane, że język może być już używany – nie ma wielkiego ryzyka, by przyjęte już standardy znacząco się zmieniły;
- dostępne są niezbędne narzędzia do budowy aplikacji XML-owych, także w systemach informacyjnych z bazami danych;
- przydatność XML w wielu dziedzinach potwierdziła się;
- XML stanowić będzie niemal na pewno ważny element systemów informacyjnych w najbliższych latach.

Wynika z tego ważny wniosek: *warto używać XML*. Więcej: już pora, by go używać!

Literatura:

- [1] T.Traczyk, W.Macewicz: „Język XML w aplikacjach z bazami danych – możliwości zastosowania, pierwsze doświadczenia”. Materiały IV Konferencji Deweloperów i Użytkowników Oracle *Ewolucja systemów informatycznych: dane, sprzęt, oprogramowanie i aplikacje*, Zakopane 1998.
- [2] T.Traczyk: „Język XML w aplikacjach z bazami danych – po roku”. Materiały V Konferencji Deweloperów i Użytkowników Oracle *Integracja danych i systemów informatycznych*, Zakopane 1999.
- [3] T.Traczyk: „Język XSL”. Materiały VI Konferencji Deweloperów i Użytkowników Oracle *Systemy informatyczne w dobie Internetu*, Zakopane 2000.
- [4] T.Traczyk: „Wprowadzenie do języka XML”. *Informatyka*, 12/1999.
- [5] T.Traczyk: „XML i XSL”. Materiały XII Górskiej Szkoły PTI, Szczyrk 2000.
- [6] Rozporządzenie Ministra Zdrowia z dn. 27 grudnia 2000 r. w sprawie trybu i sposobu przekazywania oraz zakresu danych o obrocie refundowanymi lekami i materiałami medycznymi przekazywanych przez apteki. *Dziennik Ustaw RP*, nr 4, 23 stycznia 2001, str. 287-296.
- [7] *Extensible Markup Language (XML) 1.0*. W3C Recommendation.
- [8] *Extensible Stylesheet Language (XSL) 1.0*. W3C Working Draft.
- [9] *XSL Transformations (XSLT) 1.0*. W3C Recommendation.
- [10] *Namespaces in XML*. W3C Recommendation.
- [11] *XML Schema*. W3C Candidate Recommendation.
- [12] *MSDN Online XML Developer Center*. <http://msdn.microsoft.com/xml/default.asp>
- [13] *Oracle XML-enabled*. <http://www.oracle.com/xml/>
- [14] *Oracle XML Developer's Kit*. <http://technet.oracle.com/tech/xml/>