

XML i XSL

dr inż. Tomasz Traczyk

ttraczyk@ia.pw.edu.pl

Instytut Automatyki i Informatyki Stosowanej

Politechnika Warszawska

XML jest metajęzykiem do definiowania języków znakowania. Stworzone w ten sposób języki znakowania mogą służyć np. do zapisu skomplikowanych dokumentów czy do wymiany złożonych danych. XML nie określa sposobu prezentacji dokumentów — do tego służy język XSL; XSL można też zastosować do przetwarzania dokumentów w XML.

Referat przedstawia podstawy języków XML i XSL oraz prezentuje przykłady ich zastosowań.

Wprowadzenie

Do niedawna nie istniał standardowy, powszechnie przyjęty sposób przechowywania, przesyłania i przetwarzania sformatowanych dokumentów (tekstów, raportów, arkuszy kalkulacyjnych itp.) oraz złożonych struktur danych. Stosowano dziesiątki formatów „firmowych”, co poważnie utrudniało, a niekiedy uniemożliwiało wymianę danych. Jak się wydaje, problemy te zostały przezwyciężone za sprawą języka 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 (najbardziej znanym językiem znakowania jest bez wątpienia HTML). Przetwarzanie takich plików — ze względu na to, że są to właśnie pliki tekstowe oraz że XML zaprojektowano w odpowiedni sposób — jest stosunkowo łatwe i możliwe do wykonania za pomocą standardowych, powszechnie dostępnych narzędzi.

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.

XML ma wielkie szanse na sukces ze względu na wagę problemów, do których może być stosowany, na stale rosnącą popularność WWW, a także z uwagi na poparcie deklarowane przez największych producentów oprogramowania. Rozpowszechnienie właściwego dla XML sposobu zapisu informacji może zaś mieć bardzo znaczący wpływ na rozwój całej informatyki (baz danych, architektur systemów informacyjnych itp.).

W ciągu ostatnich dwóch lat zainteresowanie językiem XML było bardzo duże. Pojawiły się pierwsze powszechnie dostępne narzędzia i pierwsze zastosowania komercyjne. Na bazie XML zdefiniowano wiele specjalizowanych języków znakowania.

Jak się wydaje, XML staje się już trwałym i ważnym składnikiem systemów informacyjnych.

Czym jest XML

XML jest metajęzykiem służącym do definiowania języków znakowania. Można w nim definiować języki opisu stron i/lub języki służące do zapisu danych wraz ze strukturą.

XML jest nieco zmodyfikowanym podzbiorem SGML. Stąd też bierze się podobieństwo do języka HTML. Cele języków są jednak zgoła odmienne: HTML jest jedynie gotowym językiem opisu stron.

Język XML ma umożliwić 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.

XML pozwala stworzyć specjalizowane języki do zapisu specyficznych typów dokumentów i umożliwia:

- łatwą wymianę danych i dokumentów między różnymi systemami informatycznymi i różnymi społecznościami użytkowników;
- tworzenie i prezentację dokumentów reprezentujących różne dziedziny za pomocą ujednoczonego oprogramowania;
- dystrybucję specjalistycznych dokumentów bezpośrednio w WWW;
- proste i ujednoczone przetwarzanie i wyszukiwanie danych;
- łatwy podział dokumentów na części i współdzielenie części (także odległych) przez różne dokumenty.

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, elementy mogą więc być zagnieżdżane.

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

DTD

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 zagnieżdżania, atrybutach elementów itp.

W XML, inaczej niż w SGML, 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.

Encje

W dokumentach XML można używać tzw. *entities* (encji, jednostek), 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 encje predefiniowane, np. `<` i `"`). 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 sformulowanym (*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 po-

siadać DTD. Dokumenty zdefiniowane jako niezależne (*standalone*), tzn. z założenia nie wyposażone w DTD, powinny być poprawne w tym sensie.

Dokumenty dobrze sformułowane mogą być zinterpretowane i wyświetlone przez przeglądarki bez konieczności dostępu do DTD. Ten stopień poprawności jest właściwy dla dokumentów, których przeznaczeniem jest jedynie wyświetlanie w przeglądarkach.

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.

Dla sprawdzenia prawidłowości niezbędne jest przypisanie dokumentu do DTD i dostępu do pliku DTD. Programy przetwarzające dane z dokumentu — zwykle budowane w oparciu o parsery języka XML — także na ogół będą wymagać dostępu do DTD.

Przykład 1

Przedstawiony poniżej przykład dokumentu w XML pochodzi z rzeczywistego systemu informacyjnego wspomagającego zarządzanie wydziałem wyższej uczelni. Konspekty opisujące zawartość wykładów sphywają do redaktora wydziałowego serwisu WWW, który przekształca je do postaci dokumentów XML. Dokumenty te są następnie wczytywane przez specjalny program do wydziałowej bazy danych, gdzie przechowywane są w postaci częściowo ustrukturalizowanej.

Oto przykładowy dokument:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<!DOCTYPE eres_konspekty SYSTEM "konspekty.dtd">
<?xml-stylesheet type="text/xsl" href="konspekty.xsl"?>
<!-- Komentarz: to jest przykład nr 1 -->
<eres_konspekty>
  <przedmiot id="KBD2" wersja="1">
    <slowo_kluczowe>bazy danych</slowo_kluczowe>
    <slowo_kluczowe>Oracle</slowo_kluczowe>
    <konspekt>
      <czesc_konspektu id="Streszczenie">
        <P> Monograficzny przedmiot poświęcony bazie danych i
          narzędziom Oracle. </P>
      </czesc_konspektu>
      <czesc_konspektu id="Treść przedmiotu">
        <P> Omawiane są podstawowe zagadnienia związane z
          wykorzystaniem RDBMS Oracle7 i <I>Oracle8</I> oraz
          administrowaniem nimi.</P>
        <P> Przedstawiane są także narzędzia do budowy aplikacji:
        </P>
        <UL>
          <LI> Oracle Forms, </LI>
          <LI> Oracle Reports. </LI>
        </UL>
      </czesc_konspektu>
    </konspekt>
  </przedmiot>
</eres_konspekty>
```

Znaczenie poszczególnych elementów dokumentu jest łatwe do zrozumienia dla każdego, kto zna podstawy HTML.

Odpowiednia definicja typu dokumentu (DTD) wygląda tak:

```
<!ELEMENT eres_konspekty      (przedmiot)+          >
<!ELEMENT przedmiot          ((slowo_kluczowe)*, konspekt) >
<!ATTLIST przedmiot id        ID #REQUIRED
```

```

wersja CDATA #IMPLIED >
<!ELEMENT slowo_kluczowe (#PCDATA) >
<!ELEMENT konspekt (czesc_konspektu)+ >
<!ELEMENT czesc_konspektu (P|UL)+ >
<!ATTLIST czesc_konspektu id ID #REQUIRED >
<!ELEMENT P (#PCDATA|I)* >
<!ELEMENT I (#PCDATA) >
<!ELEMENT UL (LI)+ >
<!ELEMENT LI (#PCDATA) >

```

W definicji elementu w nawiasach podaje się kolejne składowe — elementy podrzędne. Znak za pytania po nazwie elementu oznacza, że jest on opcjonalny. Plus po nawiasie oznacza, że elementy wymienione w nawiasie mogą się powtarzać. Pionowa kreska oznacza oczywiście alternatywę. Jeśli element nie ma mieć zawartości — tzn. jego znacznik początkowy jest jednocześnie końcowym — trzeba to zaznaczyć słowem EMPTY.

Słowo #PCDATA oznacza *parsed character data*, czyli dane znakowe, które są dalej analizowane w celu stwierdzenia, czy wewnątrz nich nie znajdują się jakieś znaczniki.

Definicje atrybutów pozwalają zdefiniować parametry elementów. Słowo #REQUIRED oznacza, że atrybut jest obowiązkowy, zaś #IMPLIED — że atrybut jest opcjonalny, ale nie ma wartości domyślnej. Po nazwie elementu i nazwie atrybutu nastąpić może lista dopuszczalnych wartości oraz wartość domyślna. Słowo ID oznacza, że atrybut jest identyfikatorem elementu; atrybuty będące odwołaniami do identyfikatorów oznacza się słowem IDREF.

XML a HTML

O szybkim zdobyciu popularności przez XML przesądziło między innymi podobieństwo dokumentów w XML do tekstów zapisanych w języku HTML. Między tymi językami są jednak znaczne różnice.

Zasadnicza różnica wynika z odmienności celów: XML ma być uniwersalnym narzędziem do zapisu różnorodnych struktur, zaś HTML ma jedynie umożliwiać zapisywanie typowych dokumentów hipertekstowych, przeznaczonych głównie do wyświetlania. Język HTML jest wprawdzie także pochodną SGML, ale nie jest jego podzbiorem, lecz konkretyzacją — odpowiada jednemu DTD. HTML zawiera zatem zamknięty zbiór znaczników, dostosowany do swego zadania. Odmiennie XML: jest on uniwersalnym narzędziem do budowy specjalizowanych języków. Nie ma zatem żadnego predefiniowanego zbioru znaczników.

Najbardziej widoczna różnica w składni języków wynika z faktu, iż zadanie przeglądarki wyświetlającej dokument jest znacznie łatwiejsze w przypadku HTML: wiedza na temat DTD dokumentu w HTML jest jakby wbudowana w przeglądarkę. W przypadku XML jest trudniej, ponieważ zestaw znaczników XML nie jest z góry zdefiniowany. Dodatkowo sprawę komplikuje założenie, iż dokument powinien dać się przeglądać nawet bez dostępu do swojego DTD. Aby przeglądarka łatwo mogła interpretować dokumenty bez DTD, tzn. poprawne w sensie *well-formed*, wprowadzono pewne obostrzenia dotyczące zamykania konstrukcji składniowych XML:

- każdy element niepusty, tzn. zaczynający się znacznikiem rozpoczynającym i zawierający jakiś tekst, musi być jawnie zakończony znacznikiem kończącym;
- elementy puste, tzn. poza ewentualnymi atrybutami nie zawierające żadnego tekstu, muszą być jawnie oznaczone jako puste.

W XML nie są zatem poprawne znane z HTML (i z SGML) struktury mające znacznik rozpoczynający, ale nie mające kończącego, np. <par>. Odpowiednik w XML musi być jawnie zakończony: <par> *treść akapitu* </par>

Podobnie, w XML nie są poprawne znane z HTML puste elementy, np.
. Takie elementy muszą być jawnie oznaczone jako puste przez wstawienie ukośnika na końcu znacznika; odpowiednikiem w XML jest zatem
.

Dla autora piszącego tekst dokumentu te surowe reguły zamykania struktur mogą być lekko irytujące, ale za to dla programu przetwarzającego taki dokument stanowi to istotne ułatwienie. Struktura dokumentu staje się też znacznie bardziej przejrzysta, nawet dla kogoś, kto nie zna DTD ani semantyki poszczególnych znaczników.

Z pozostałych różnic składniowych między XML i HTML najbardziej znaczące wydają się: wrażliwość XML na wielkość liter oraz obowiązek ujmowania wartości atrybutów w cudzysłowach.

Inne składniki XML

Rozwój zastosowań XML spowodował potrzebę uzupełnienia języka o dodatkowe składniki. Opisano tu najważniejsze.

Przestrzenie nazw

Można się spodziewać, że po rozpowszechnieniu XML powstanie bardzo wiele systemów (słowników) znaczników, przeznaczonych do stosowania w swych specyficznych dziedzinach. Autor, budujący dokument z danej dziedziny, powinien korzystać z już istniejących systemów znaczników, ale może potrzebować znaczników z więcej niż jednego słownika, może także chcieć dołączyć swoje własne znakowanie. W takiej, bardzo przecież typowej, sytuacji może dojść do konfliktów nazw: nazwy znaczników i atrybutów zdefiniowane w różnych słownikach mogą się pokrywać.

Aby zapobiec tego typu problemom, określono sposób wyznaczania i wykorzystania tzw. przestrzeni nazw (*XML namespaces*) [9].

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, a nawet w jednym elemencie, można powołać się na wiele przestrzeni nazw.

W dokumencie z przykładu 1 można by użyć specjalnej przestrzeni nazw tak:

```
<eres:konspekty xmlns:eres="http://www.elka.pw.edu.pl/eres">
  <eres:slovo_kluczowe>
    ...
```

Zdefiniowano tu prefiks `eres` i przypisano go do przestrzeni nazw, a następnie wykorzystano ten prefiks do wyróżnienia znaczników należących do powołanej przestrzeni nazw.

Schematy i typy danych

DTD określa precyzyjnie składnię znaczników, ale nie daje pełnych możliwości potrzebnych przy reprezentowaniu w XML złożonych danych i ich automatycznym przetwarzaniu. DTD nie pozwala bowiem precyzyjnie określić typów danych, struktura dokumentu wyrażona przez DTD nie jest podatna na rozbudowę, DTD nie objaśnia znaczenia elementów, nie pozwala na określenie ograniczeń ani zależności referencyjnych między elementami.

Problemy te starają się rozwiązać propozycje, wprowadzające zamiast DTD tzw. schematy (*XML Schema* [10]). Budowa dokumentów jest tu wyrażona z użyciem składni samego XML, bez konieczności odwoływania się do składni DTD. Struktura elementów dokumentu XML jest traktowana tak jak hierarchia klas obiektów. Schematy pozwalają na zdefiniowanie m.in. typów danych, opisów znaczenia klas i atrybutów, ograniczeń, związków typu referencyjnego. Możliwy jest import do schematu fragmentów definicji innych schematów, co pozwala na sformalizowane rozbudowywanie definicji dokumentów.

Zastosowanie schematów może pozwolić aplikacjom na szczegółowe — nie ograniczające się jedynie do składni znaczników — sprawdzanie poprawności zawartości dokumentów, w tym poprawności typów danych.

Łączniki

Dokument w XML może zawierać łączniki — odwołania do innych dokumentów.

Łączniki definiuje się w języku XML (*XML Linking Language*) [5]. Zasadniczą część definicji łączników stanowią tzw. lokatory. Do definiowania lokatorów używa się adresów zwanych URI (*Uniform Resource Identifier*), stanowiących rozszerzenie URL. URI zawierać może adres sieciowy, pytanie (*query*, po znaku ?) oraz identyfikator fragmentu (*fragment identifier*, po znaku #). Możliwości łączenia dokumentów są w XML znacznie bogatsze od znanych z HTML.

Sposób adresowania miejsc i fragmentów wewnątrz dokumentów określa specyfikacja XPointer (*XML Pointer Language*) [6]. Wskazanie fragmentu dokumentu może być bardziej skomplikowane niż w HTML i nie jest do tego niezbędne umieszczenie w tym dokumencie żadnego specjalnego oznaczenia.

Adresowanie w języku XPointer opiera się na drzewie elementów — na ich hierarchii i kolejności. Możliwe jest także użycie identyfikatorów elementów. Do precyzyjnego wskazania położenia w dokumencie służą specjalne wyrażenia-adresy zdefiniowane w standardzie XPath [11]. Typowe adresy XPath wskazują na określone wystąpienie konkretnego typu elementu w danym kontekście, np. można wskazać na trzeci element składowy drugiego elementu określonego typu, następującego po podanym identyfikatorze.

Jeśli adres URI wskazuje na wnętrze dokumentu w XML, to zawarty w URI identyfikator fragmentu powinien być sformułowany zgodnie ze specyfikacją XPointer.

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 na ekranie lub jego druk. Inne typowe rodzaje przetwarzania to wyszukiwanie danych w dokumencie lub zbiorze dokumentów i wykorzystanie danych z dokumentu w programach.

Ponieważ te rodzaje przetwarzania są powszechne, opracowano 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 służy język XSL (*eXtensible Stylesheet Language*). Możliwe jest także definiowanie prezentacji dokumentów w XML za pomocą języka CSS.

DOM i SAX

Dane zawarte w dokumentach XML powinny dać się wygodnie przetwarzać w programach. Aby z zastosowania XML płynęła jakaś korzyść dla programistów tworzących takie programy, potrzebny jest 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 języków programowania Java, ECMAScript, VBScript i C++.

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), umożliwiając wygodne manipulowanie dokumentami w typowych środowiskach programowania. Najbardziej znana implementacja DOM jest częścią MSIE 5.0.

Pewną popularność zdobył także, spełniający podobne jak DOM zadanie, model SAX (*Simple API for XML*), dla którego dostępne są darmowe parsery, np. dla języka Java.

SAX opiera się na modelu programowania zdarzeniowego: 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.

XQL

Typową operacją wykonywaną na dokumentach i danych jest wyszukiwanie. Potrzebny jest zatem standardowy sposób zadawania warunków wyszukiwania. Pojawiła się więc propozycja stworzenia specjalnego języka zapytań XQL (*XML Query Language*).

XQL ma umożliwiać wyszukiwanie danych w dokumencie lub kolekcji dokumentów (np. repozytorium). Zapytanie w XQL ma bardzo prostą budowę i jest pozbawione „ozdobnych” słów kluczowych (typu ‘select’), gdyż ma dawać użyć się jako *fragment identifier* w adresie URI.

Język XSL

XSL (*eXtensible Stylesheet Language*) [7] jest językiem służącym do prezentacji dokumentów w XML i do ich przekształcania.

Składnia XSL została w pełni zdefiniowana w XML, z użyciem przestrzeni nazw (*namespaces*). XSL składa się z dwóch części: języka transformacji XSLT (*XSL Transformations*) [8] oraz słownika obiektów specyfikujących formatowanie dokumentu (*formatting objects*).

Prezentacja dokumentu z użyciem XSL przebiega w dwóch fazach: transformacji hierarchii znaczników na hierarchię tzw. *flow objects* i przypisania tym obiektom 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, będą zapewne także przystosowane do prezentacji HTML, wygodnym sposobem formatowania dokumentów za pomocą XSL jest przetworzenie ich na HTML. Tak też skonstruowany został poniższy przykład.

Przykład 2

Poniższy skrypt w XSL służy do prezentacji konspektu dokumentu z przykładu 1.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

  <xsl:template match="/">
    <HTML>
      <HEAD><TITLE>Konspekty</TITLE></HEAD>
      <BODY>
        <H1>Konspekty przedmiotów</H1>
        <xsl:apply-templates select="eres_konspekty/przedmiot"
                           order-by="@id"/>
      </BODY>
    </HTML>
  </xsl:template>

  <xsl:template match="przedmiot">
    <H2><xsl:value-of select="@id"/>
    (<xsl:value-of select="@wersja"/>)</H2>
  </xsl:template>
</xsl:stylesheet>
```

```

    <H3>Słowa kluczowe</H3>
    <TABLE BORDER="1">
    <xsl:for-each select="słowo_kluczowe" order-by="text()">
      <TR><TD><xsl:value-of /></TD></TR>
    </xsl:for-each>
    </TABLE>
    <xsl:apply-templates/>
    <HR/>
  </xsl:template>

  <xsl:template match="konspekt">
    <H3>Konspekt</H3>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="czesc_konspektu">
    <H4><xsl:value-of select="@id" /></H4>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="P">
    <P><xsl:apply-templates /></P>
  </xsl:template>

  <xsl:template match="I">
    <I><xsl:apply-templates /></I>
  </xsl:template>

  <xsl:template match="UL">
    <UL><xsl:apply-templates /></UL>
  </xsl:template>

  <xsl:template match="LI">
    <LI><xsl:apply-templates /></LI>
  </xsl:template>

  <xsl:template match="text()">
    <xsl:value-of />
  </xsl:template>
</xsl:stylesheet>

```

Istotę działania XSL stanowi rekurencyjne przetwarzanie znaczników. Fraza `<xsl:apply-templates/>` jest wywołaniem owej rekurencji. Klauzule `select` zawężają zakres działania do wybranych elementów dokumentu. Klauzule `match` podają wzorzec znacznika, który jest przetwarzany za pomocą danej frazy `template`. Do definiowania wzorców służą wyrażenia zgodne ze specyfikacją XPath [11], tą samą, której używa się w języku XPointer.

Frazy `value-of` powodują włączenie odpowiedniej części przetwarzanego dokumentu do dokumentu wynikowego. Nazwy poprzedzone znakiem `@` oznaczają odwołanie do wartości atrybutu.

Efekty formatowania przykładowego dokumentu za pomocą powyższego skryptu przedstawia Rysunek 1.



Rysunek 1. Wynik formatowania dokumentu XML za pomocą XSL

Zastosowania XML

W ciągu ostatniego roku język XML znacznie zyskał na popularności, pojawiło się bardzo wiele propozycji jego wykorzystania oraz sporo narzędzi wspomagających użycie XML.

Podstawowe narzędzia

Dostępne obecnie narzędzia dla XML to:

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

Najbardziej znanym produktem jest Microsoft Internet Explorer 5.0, zawierający obsługę modelu DOM oraz umożliwiający walidację dokumentów z użyciem DTD. MSIE potrafi prezentować dokumenty XML z wykorzystaniem arkuszy stylistycznych napisanych w językach XSL (zrealizowano jedynie pierwszą część przetwarzania — transformację, i to według starszej wersji specyfikacji XSL, bez rozszerzeń wprowadzonych w XSLT) oraz CSS.

Do prezentowania dokumentów w XML z użyciem CSS dostosowano także najnowszą wersję 6 przeglądarki firmy Netscape. Niestety, w dostępnej obecnie wersji próbnej brak obsługi języka XSL.

Wykorzystanie w programach powszechnego użytku

XML uzyskał poparcie znaczących producentów oprogramowania i już jest używany w programach powszechnego użytku. Zapewne najbardziej znanym przykładem jest pakiet Microsoft Office 2000.

W pakiecie tym umożliwiono zapis większości dokumentów (np. złożonych dokumentów tekstowych czy arkuszy kalkulacyjnych) w formacie HTML. Należało jednak zapewnić odwracalność zapisu, tzn. możliwość ponownego otworzenia przez programy MS Office tak zapisanego dokumentu bez utraty informacji. Możliwości HTML do tego nie wystarczają, zapis w HTML uzupełniono więc właśnie językiem XML. Zapisane w taki sposób dokumenty mogą być dystrybuowane w WWW, z zachowaniem możliwości ich pełnego odtworzenia do dalszej edycji. Dodatkowo rozwiązanie takie zapewnia możliwość wymiany dokumentów między różnymi wersjami oprogramowania — dokumenty z przyszłych wersji MS Office będą mogły być prawidłowo czytane przez wersje wcześniejsze, gdyż znaczniki niezrozumiałe dla starszych wersji będą po prostu ignorowane.

Ciekawym zastosowaniem XML, także związanym z MS Office, ale popieranym i przez innych producentów, jest język VML (*Vector Markup Language*). Służy on do definiowania i prezentowania grafiki wektorowej, a zapisywane są w nim m.in. rysunki OfficeArt — wykonywane za pomocą programów pakietu MS Office. Przeglądarka MSIE 5.0 potrafi bezpośrednio wyświetlać takie rysunki. W3C prowadzi prace standaryzacyjne, mające na celu scalenie kilku istniejących propozycji specyfikacji grafiki wektorowej w jeden standard o nazwie *Scalable Vector Graphics* (SVG).

Specjalizowane struktury danych

Powstaje 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*);
- różne projekty z dziedziny EDI (*Electronic Data Interchange*);
- finanse i bankowość, np. OFX/OFE (*Open Financial Exchange*), BIPS (*Bank Internet Payment System*);
- multimedia, np. SMIL (*Synchronized Multimedia Integration Language*), PGML (*Precision Graphics Markup Language*), czy wspomniany już VML.

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.

Pierwszy krok stanowi skonstruowanie interfejsów umożliwiających przekształcanie informacji zgromadzonej w bazach danych w postaci relacyjnej na język XML i odwrotnie. To zagadnienie jest stosunkowo proste i pierwsze rozwiązania komercyjne, np. parsery XML zintegrowane z systemami zarządzania bazami danych, już się pojawiają.

Znacznie trudniejsze problemy przyniesie może upowszechnienie XML-owego podejścia do reprezentowania złożonej informacji. Często natrafi się wówczas na struktury, które trudno jest przekształcić do postaci relacyjnej lub ich reprezentacja w pełni relacyjna jest nieefektywna (dokument z przykładu 1 ilustruje ten problem). Konieczne jest więc znalezienie nowych sposobów przechowywania takiej informacji w bazach danych, z zachowaniem możliwości efektywnego wyszukiwania informacji.

Obecnie podejmowane są próby przechowywania dokumentów w formie częściowo ustrukturalizowanej, ale na ogół nie spełniają one wymagań efektywności. Upowszechnienie XML może więc spowodować znaczne zmiany w technologii samych baz danych i w sposobach ich stosowania.

Podsumowanie

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 specyfikacja języka XML, przestrzeni nazw, XSLT, XPath oraz modelu DOM mają status *W3C Recommendation*, oznaczający ostatnią fazę przed uznaniem standardu. Mniej zaawansowane są prace nad specyfikacjami schematów, XSL, XLL i XPointer — mają status *W3C Working Draft*.

XML przeżywa obecnie szybki rozwój. Specyfikacje podlegają szybkiej ewolucji, a na ustabilizowanie się standardu przyjdzie zapewne jeszcze poczekać. Pojawienie się na rynku pierwszych po-

wszechnie dostępnych narzędzi tworzy jednak pewien standard *de facto*, do którego zapewne będą musiały być dostosowywane formalne definicje.

Rola XML

Język XML zdobywa coraz szersze uznanie i okazuje się przydatny w licznych zastosowaniach z wielu różnych dziedzin. Język zyskał poparcie wszystkich znaczących producentów oprogramowania. Jego specyfikacja jest nadal udoskonalana, dodawane są też nowe istotne komponenty.

XML staje się nowym standardem zapisu i przekazywania informacji. Wydaje się atrakcyjny dla bardzo różnych grup użytkowników. Wykorzystanie XML stanie się zapewne już w najbliższych latach powszechne. Wpłynie to na pewno znacząco na rozwój technologii systemów informacyjnych, a w szczególności na systemy internetowe i systemy z bazami danych.

Literatura:

- [1] T.Traczyk, W.Macewicz: „Język XML w aplikacjach z bazami danych — możliwości zastosowania, pierwsze doświadczenia”. Materiały IV Konferencji Developerów 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 Developerów i użytkowników Oracle *Integracja danych i systemów informatycznych*, Zakopane 1999.
- [3] T.Traczyk: „Wprowadzenie do języka XML”. *Informatyka*, 12/1999.
- [4] *Extensible Markup Language (XML) 1.0*. W3C Recommendation.
- [5] *XML Linking Language (XLink)*. W3C Working Draft.
- [6] *XML Pointer Language (XPointer)*. W3C Working Draft.
- [7] *Extensible Stylesheet Language (XSL) 1.0*. W3C Working Draft.
- [8] *XSL Transformations (XSLT) 1.0*. W3C Recommendation.
- [9] *Namespaces in XML*. W3C Recommendation.
- [10] *XML Schema*. W3C Working Draft.
- [11] *XML Path Language (XPath) 1.0*. W3C Recommendation.