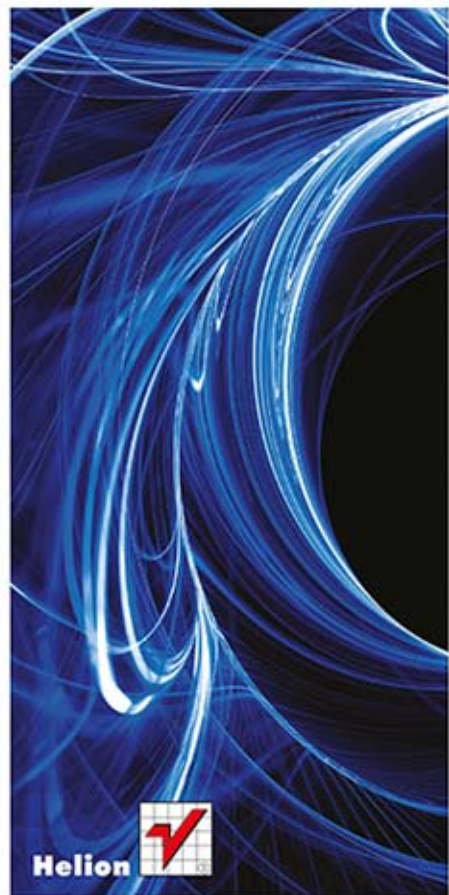


SZYBKI START



XML

WYDANIE II



Helion



XML • root element • well-formed • XSL • XSLT • node sets • XPath
entities • XML Schema • simple types • complex types • namespace • va
XQuery • XML • root element • well-formed • XSL • XSLT • node sets
DTD • entities • XML Schema • simple types • complex types • nam
validation • XQuery • XML • root element • well-formed • XSL
sets • XPath • DTD • entities • XML Schema • simple types • complex
namespace • validation • XQuery • XML • root element • well-forme
XSLT • node sets • XPath • DTD • entities • XML Schema • simple
complex types • namespace • validation • XQuery • XML • root el
well-formed • XSL • XSLT • node sets • XPath • DTD • entities • XML
simple types • complex types • namespace • validation • XQuery
root element • well-formed • XSL • XSLT • node sets • XPath • DTD • e
XML Schema • simple types • complex types • namespace • validatio
well-formed • XSL • XSLT • node sets • XPath • DTD • entities • XML
simple types • complex types • namespace • validation • XML el
XSL • XSLT • node sets • XPath • DTD • entities • XML Schema • simple
complex types • namespace • validation • XQuery • XML • root elemen
formed • XSL • XSLT • node sets • XPath • DTD • entities • XML Schema
types • complex types • namespace • validation • XQuery • XML • root e
well-formed • XSL • XSLT • node sets • XPath • DTD • entities • XML
simple types • complex types • namespace • validation • XQuery
root element • well-formed • XSL • XSLT • node sets • XPath • DTD •
XML Schema • simple types • complex types • namespace • vali
XQuery • XML • root element • well-formed • XSL • XSLT • node sets
DTD • entities • XML Schema • simple types • complex types • nam
validation • XQuery • XML • root element • well-formed • XSL • XSLT
sets • XPath • DTD • entities • XML Schema • simple types • complex
namespace • validation • XQuery • XML • root element • well-forme
XSL • XSLT • node sets • XPath • DTD • entities • XML Schema • simple
complex types • namespace • validation • XQuery • XML • root elemen

**Przewodnik
po świecie XML!**

KEVIN HOWARD GOLDBERG

Tytuł oryginału: XML Visual QuickStart Guide (2nd Edition)

Tłumaczenie: Lech Lachowski

ISBN: 978-83-246-8237-9

Authorized translation from the English language edition, entitled: XML: VISUAL QUICKSTART GUIDE, Second Edition; ISBN 0321559673; by Kevin Howard Goldberg; published by Pearson Education, Inc, publishing as Peachpit Press. Copyright © 2009 by Kevin Howard Goldberg and Elizabeth Castro.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Polish language edition published by HELION S.A., Copyright © 2014.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/xmlss2>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<ftp://ftp.helion.pl/przyklady/xmlss2.zip>

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

	Wprowadzenie	II
Część I	XML	19
Rozdział 1.	Pisanie dokumentów XML	21
	Przykładowy dokument XML	22
	Zasady pisania dokumentów XML	23
	Elementy, atrybuty i wartości	24
	Jak zacząć	25
	Tworzenie elementu głównego	26
	Pisanie elementów-dzieci	27
	Zagnieżdżanie elementów	28
	Dodawanie atrybutów	29
	Stosowanie pustych elementów	30
	Pisanie komentarzy	31
	Encje predefiniowane — pięć znaków specjalnych	32
	Wyświetlanie elementów w postaci tekstu	33
Część II	XSL	35
Rozdział 2.	XSLT	37
	Przekształcanie dokumentów XML za pomocą XSLT	38
	Inicjowanie arkusza stylów XSLT	40
	Tworzenie szablonu głównego	41
	Uzyskiwanie pliku wyjściowego HTML	42
	Wyświetlanie wartości	44
	Zapętlanie węzłów	46
	Warunkowe przetwarzanie węzłów	48
	Dodawanie wyborów warunkowych	49
	Sortowanie węzłów przed przetworzeniem	50
	Generowanie atrybutów wyjściowych	51
	Tworzenie i stosowanie szablonów	52

Rozdział 3.	Wzorce i wyrażenia XPath	55
	Lokalizowanie węzłów	56
	Ustalanie bieżącego węzła	58
	Odwoływanie się do węzła bieżącego	59
	Wybieranie dzieci danego węzła	60
	Wybieranie rodzica lub rodzeństwa danego węzła	61
	Wybieranie atrybutów węzła	62
	Warunkowe wybieranie węzłów	63
	Tworzenie bezwzględnych ścieżek lokalizacji	64
	Wybieranie wszystkich potomków	65
Rozdział 4.	Funkcje XPath	67
	Porównywanie dwóch wartości	68
	Testowanie pozycji	69
	Mnożenie, dzielenie, dodawanie i odejmowanie	70
	Liczenie węzłów	71
	Formatowanie liczb	72
	Zaokrąglanie liczb	73
	Wyodrębnianie podciągów	74
	Zmianie wielkości znaków w ciągu	75
	Sumowanie wartości	76
	Więcej funkcji języka XPath	77
Rozdział 5.	XSL-FO	79
	Dwie części dokumentu XSL-FO	80
	Tworzenie dokumentu XSL-FO	81
	Tworzenie i stylizowanie bloków zawartości strony	82
	Dodawanie obrazków	83
	Definiowanie szablonu strony	84
	Tworzenie nagłówka szablonu strony	85
	Tworzenie dokumentów XSL-FO za pomocą XSLT	86
	Wstawianie podziałów stron	87
	Wyświetlanie zawartości strony w kolumnach	88
	Dodawanie szablonu nowej strony	89
Część III	DTD	91
Rozdział 6.	Tworzenie dokumentów DTD	93
	Praca z dokumentami DTD	94
	Definiowanie elementu zawierającego tekst	95
	Definiowanie pustego elementu	96
	Definiowanie elementu, który zawiera element-dziecko	97
	Definiowanie elementu, który zawiera kilka elementów-dzieci	98
	Definiowanie liczby wystąpień	99

	Definiowanie wyborów	100
	Definiowanie elementu, który ma dowolną zawartość	101
	Kilka słów o atrybutach	102
	Definiowanie atrybutów	103
	Definiowanie wartości domyślnych	104
	Definiowanie atrybutów z wyborami	105
	Definiowanie atrybutów z wartościami unikatowymi	106
	Odwolywanie się do atrybutów z wartościami unikatowymi	107
	Ograniczanie atrybutów do poprawnych nazw XML	108
Rozdział 7.	Encje i notacje w dokumentach DTD	109
	Tworzenie encji ogólnej	110
	Korzystanie z encji ogólnych	111
	Tworzenie zewnętrznej encji ogólnej	112
	Korzystanie z zewnętrznych encji ogólnych	113
	Tworzenie encji dla zawartości nieparsowanej	114
	Wstawianie zawartości nieparsowanej	116
	Tworzenie i wykorzystywanie encji parametrycznych	118
	Tworzenie zewnętrznej encji parametrycznej	119
Rozdział 8.	Walidacja i używanie DTD	121
	Tworzenie zewnętrznego dokumentu DTD	122
	Deklarowanie zewnętrznego DTD	123
	Deklarowanie i tworzenie wewnętrznego DTD	124
	Walidacja dokumentów XML względem DTD	125
	Nazywanie publicznego zewnętrznego DTD	126
	Deklarowanie publicznego zewnętrznego DTD	127
	Wady i zalety DTD	128
Część IV	XML Schema	129
Rozdział 9.	Podstawy XML Schema	131
	Praca z XML Schema	132
	Rozpoczynanie prostego dokumentu XML Schema	134
	Wiązanie XML Schema z dokumentem XML	135
	Dodawanie adnotacji do schematów	136
Rozdział 10.	Definiowanie typów prostych	137
	Definiowanie elementu jako typu prostego	138
	Używanie typów daty i czasu	140
	Używanie typów liczbowych	142
	Predefiniowanie zawartości elementu	143
	Wywodzenie niestandardowych typów prostych	144
	Wywodzenie nazwanych typów niestandardowych	145

Określanie zakresu dopuszczalnych wartości	146
Określanie zbioru dopuszczalnych wartości	148
Ograniczanie długości elementu	149
Określanie wzorca dla elementu	150
Ustawianie limitu cyfr w liczbie	152
Wywodzenie typu listy	153
Wywodzenie typu łączonego	154

Rozdział 11. **Definiowanie typów złożonych** **155**

Podstawy typów złożonych	156
Wywodzenie anonimowych typów złożonych	158
Wywodzenie nazwanych typów złożonych	159
Definiowanie typów złożonych, które zawierają elementy-dzieci	160
Ustalanie określonej sekwencji dla występowania elementów-dzieci	161
Dopuszczanie, aby elementy-dzieci pojawiały się w dowolnej kolejności	162
Tworzenie zbioru wyborów	163
Definiowanie elementów zawierających wyłącznie tekst	164
Definiowanie pustych elementów	165
Definiowanie elementów z zawartością mieszaną	166
Wywodzenie typów złożonych z innych istniejących typów złożonych	167
Odwolywanie się do elementów zdefiniowanych globalnie	168
Kontrolowanie liczby wystąpień	169
Definiowanie nazwanych grup modelowych	170
Odwolywanie się do nazwanej grupy modelowej	171
Definiowanie atrybutów	172
Wymaganie atrybutu	173
Predefiniowanie zawartości atrybutu	174
Definiowanie grup atrybutów	175
Odwolywanie się do grup atrybutów	176
Lokalne i globalne definicje	177

Część V **Przestrzenie nazw** **179**

Rozdział 12. **Przestrzenie nazw XML** **181**

Projektowanie nazwy przestrzeni nazw	182
Deklarowanie domyślnej przestrzeni nazw	183
Deklarowanie prefiksu nazwy przestrzeni nazw	184
Oznaczanie elementów prefiksem przestrzeni nazw	185
W jaki sposób przestrzenie nazw wpływają na atrybuty	186

Rozdział 13. **Używanie przestrzeni nazw XML** **187**

Wypełnianie przestrzeni nazw XML	188
Dokumenty XML Schema, dokumenty XML oraz przestrzenie nazw	189
Odwolywanie się do komponentów XML Schema w przestrzeniach nazw	190



Przestrzenie nazw i walidacja dokumentów XML	191
Dodawanie wszystkich elementów zdefiniowanych lokalnie	192
Dodawanie poszczególnych elementów zdefiniowanych lokalnie	193
Definicje XML Schema w kilku plikach	194
Dokumenty XML Schema z kilkoma przestrzeniami nazw	195
Schemat schematów jako ustawienie domyślne	196
Przestrzenie nazw i dokumenty DTD	197
XSLT i przestrzenie nazw	198

Część VI **Najnowsze rekomendacje W3C** **199**

Rozdział 14. **XSLT 2.0** **201**

Rozszerzanie XSLT	202
Tworzenie uproszczonego arkusza stylów	203
Generowanie dokumentów wynikowych XHTML	204
Generowanie wielu dokumentów wynikowych	205
Tworzenie funkcji definiowanych przez użytkownika	206
Wywoływanie funkcji definiowanych przez użytkownika	207
Grupowanie danych wynikowych na podstawie wspólnych wartości	208
Walidacja plików wynikowych XSLT	209

Rozdział 15. **XPath 2.0** **211**

XPath 1.0 i XPath 2.0	212
Uśrednianie wartości w sekwencji	214
Wyznaczanie wartości minimalnej lub maksymalnej	215
Formatowanie ciągów znaków	216
Testowanie warunków	217
Kwantyfikacja warunku	218
Usuwanie zduplikowanych pozycji	219
Zapętlanie sekwencji	220
Używanie bieżącej daty i czasu	221
Pisanie komentarzy	222
Przetwarzanie danych wejściowych typu nie-XML	223

Rozdział 16. **XQuery 1.0** **225**

XQuery 1.0 a XSLT 2.0	226
Redagowanie dokumentu XQuery	227
Identyfikowanie dokumentu źródłowego XML	228
Używanie wyrażeń ścieżkowych	229
Pisanie wyrażeń FLWOR	230
Testowanie za pomocą wyrażeń warunkowych	232
Łączenie dwóch powiązanych źródeł danych	233
Tworzenie i wywoływanie funkcji definiowanych przez użytkownika	234
XQuery i bazy danych	235

Część VII	XML w praktyce	237
Rozdział 17.	Ajax, RSS, SOAP i inne zastosowania języka XML	239
	Podstawy technologii Ajax	240
	Przykłady zastosowania techniki Ajax	242
	Podstawy RSS	245
	Schematy RSS	246
	Rozszerzanie RSS	247
	SOAP i usługi internetowe	249
	Schemat komunikatu SOAP	250
	WSDL	251
	Podstawy KML	253
	Prosty plik KML	254
	ODF i OOXML	255
	eBook, ePub itp.	257
	Narzędzia dla XML-a w praktyce	259
	Dodatki	261
Dodatek A	Narzędzia XML	263
	Edytory XML	264
	Dodatkowe edytory XML	266
	Narzędzia i zasoby XML	267
Dodatek B	Zestawy znaków i encje	269
	Określanie sposobu kodowania znaków	270
	Używanie numerycznych odwołań znakowych	271
	Korzystanie z referencji encji	272
	Znaki Unicode	273
	Skorowidz	275

W części I tej książki zapoznałeś się z bazową gramatyką XML-a, która określa reguły pisania dokumentów XML. W części II nauczyłeś się przekształcać dokumenty XML do innej postaci, którą w naszym przypadku były pliki HTML. Z kolei w części III dowiesz się, jak zdefiniować niestandardowy język znaczników w XML-u.

Aby zdefiniować taki język, musisz najpierw określić jego elementy oraz ich atrybuty, a także zadeklarować, które są wymagane, a które nie. Takie informacje zwane są **schematem** (ang. *schema*). Historyk mógłby na przykład utworzyć język CsśML, (fikcyjny) język znaczników cudów starożytnego świata, będący systemem katalogowania danych dotyczących cudów starożytności. CsśML mógłby zawierać takie elementy jak `cud`, `nazwa`, `rok_wybudowania` oraz `dzieje`.

Schematy, chociaż nie są wymagane, są niezwykle istotne dla zachowania spójności dokumentów XML. W rzeczywistości możesz porównać dowolny dokument XML z odpowiadającym mu schematem, aby *zaweryfikować*, czy jest on zgodny z regułami zdefiniowanymi w tym schemacie (patrz rozdział 8.). Jeśli taki dokument XML zostanie uznany za poprawny, oznacza to, że jego dane mają właściwą formę — taką, jaka została określona w schemacie.

Istnieją dwa zasadnicze systemy pisania schematów: DTD oraz XML Schema. DTD, czyli **definicja typu dokumentu** (ang. *Document Type Definition*), jest starszym ale szeroko stosowanym systemem ze specyficzną i ograniczoną składnią. Kolejne trzy rozdziały poświęcone są pisaniu schematów w stylu DTD. Drugi podstawowy system, XML Schema, został opisany szczegółowo w części IV tej książki. Z różnych powodów możesz preferować jeden lub drugi system (patrz podrozdział „Wady i zalety DTD” w rozdziale 8.).

Praca z dokumentami DTD

DTD, czyli **definicja typu dokumentu** (ang. *Document Type Definition*), jest zbiorem reguł, które definiują niestandardowy język znaczników w XML-u. DTD w swej istocie po prostu identyfikuje elementy i ich atrybuty. Jeśli dokument XML nie stosuje się do reguł zdefiniowanych przez DTD, nie jest uznawany za poprawny dla tego konkretnego języka niestandardowego. Za pomocą takiego testu walidacji możesz szybko rozpoznać, czy dany dokument XML przestrzega zasad, które zdefiniowałeś dla swojego języka, czy też nie.

Jak już wspomniano, XML wykorzystuje te same bloki składowe co HTML: elementy, atrybuty oraz wartości. Elementy są fundamentalnymi jednostkami dokumentu XML (rysunek 6.1). Mogą one przyjmować wartości, posiadać atrybuty oraz zawierać inne elementy. Schemat DTD dla danego języka znaczników niestandardowych będzie definiował listę elementów oraz wszelkich elementów-dzieci, które każdy element może mieć (rysunek 6.2). Będzie też definiował wszelkie atrybuty, które każdy element może posiadać, oraz będzie definiował, czy te elementy i atrybuty są opcjonalne czy wymagane. W ten sposób DTD definiuje prawidłową strukturę języka znaczników niestandardowych, a zatem również dowolny poprawny dokument XML, który jest częścią tego języka.

DTD jest dokumentem wyłącznie tekstowym i zwyczajowo jest zapisywany z rozszerzeniem *.dtd*. Nie jest on dokumentem XML i dlatego nie rozpoczyna się od standardowej deklaracji XML.

Wskazówki

- ✓ Zastosowanie dokumentów DTD jest doskonałym sposobem zapewnienia spójności danych XML udostępnianych różnym osobom i firmom. Przed użyciem dokumentów XML otrzymywanych od innych osób możesz wykorzystywać DTD do sprawdzania, czy dokumenty te mają właściwy format.
- ✓ Do sprawdzenia zgodności dokumentu XML z danym dokumentem DTD będziesz potrzebował edytora XML lub jakiegoś procesora DTD. Szczegółowe informacje na temat tych dwóch typów narzędzi znajdziesz w dodatku A.

xml
<pre><?xml version="1.0"?> <cud> <nazwa>Kolos Rodyjski</nazwa> <lokalizacja>Grecja</lokalizacja> <wysokosc>33</wysokosc> </cud></pre>

Rysunek 6.1. Oto jeden z pierwszych dokumentów XML, które napotkałeś w tej książce. Składa się z czterech elementów: elementu głównego o nazwie *cud* oraz trzech elementów-dzieci (*nazwa*, *lokalizacja* i *wysokosc*)

dtd
<pre><!ELEMENT cud (nazwa, lokalizacja, wysokosc)> <!ELEMENT nazwa (#PCDATA)> <!ELEMENT lokalizacja (#PCDATA)> <!ELEMENT wysokosc (#PCDATA)></pre>

Rysunek 6.2. Ten fragment kodu DTD definiuje strukturę dokumentu XML pokazanego na rysunku 6.1. Można go odczytać następująco. Element *cud* zawiera trzy elementy-dzieci: *nazwa*, *lokalizacja* i *wysokosc*. Wszystkie są elementami typu *PCDATA*. *PCDATA* zostało omówione szczegółowo w podrozdziale „Definiowanie elementu zawierającego tekst”. Teraz powiem tylko, że *PCDATA* to po prostu tekst

```

xml
<?xml version="1.0"?>
<cuda_starozytnosci>
  <cuda>
    <nazwa jezyk="polski">Kolos Rodyjski</nazwa>
    <nazwa jezyk="grecki">Κολοσσός της Ρόδου</nazwa>
    <lokalizacja>Rodos, Grecja</lokalizacja>
    <wysokosc jednostka="metr">33</wysokosc>
    <historia>
      <rok_wybudowania era="p.n.e">282
        </rok_wybudowania>
      <rok_zniszczenia era="p.n.e.">226
        </rok_zniszczenia>
      <sposob_zniszczenia>trzęsienie ziemi
        </sposob_zniszczenia>
      <dzieje>W roku 294 p.n.e. mieszkańcy wyspy
        Rodos ...</dzieje>
    </historia>
  </cuda>
</cuda_starozytnosci>

```

Rysunek 6.3. *Zwróć uwagę, że w tej uproszczonej wersji dokumentu głównego XML większość elementów zawiera tekst. (Niektóre z nich zawierają również atrybuty, które zostały omówione w podrozdziale „Na temat atrybutów”). Elementy-dzieci elementu historia również zawierają tekst, przy czym sam element historia tekstu nie zawiera. Elementy zawierające elementy-dzieci zostały omówione w podrozdziale „Definiowanie elementu, który zawiera element-dziecko”*

```

dtd
<!ELEMENT nazwa (#PCDATA)>
<!ELEMENT lokalizacja (#PCDATA)>
<!ELEMENT wysokosc (#PCDATA)>
<!ELEMENT rok_wybudowania (#PCDATA)>
<!ELEMENT rok_zniszczenia (#PCDATA)>
<!ELEMENT sposob_zniszczenia (#PCDATA)>
<!ELEMENT dzieje (#PCDATA)>

```

Rysunek 6.4. *Prawie każdy dokument DTD będzie zawierał elementy, które zostały zdefiniowane jako PCDATA. W tym fragmencie kodu DTD pokazałem wszystkie elementy z rysunku 6.3, które zostały zdefiniowane jako elementy zawierające tekst*

Definiowanie elementu zawierającego tekst

Aby zdefiniować w DTD strukturę własnego języka znaczników niestandardowych, powinieneś zdefiniować strukturę i zawartość elementów, które posiadałyby poprawny dokument XML.

Wiele elementów w Twoim dokumencie XML będzie zawierać po prostu tekst (rysunek 6.3). O ile element adres może zawierać elementy potomne, takie jak ulica, miasto, województwo i kod_pocztowy, to już te elementy prawdopodobnie będą zawierać jedynie tekst.

Aby zdefiniować element, który zawiera wyłącznie tekst:

1. Wpisz `<!ELEMENT znacznik`, gdzie *znacznik* jest nazwą elementu, który chcesz zdefiniować.
2. Następnie wpisz `(#PCDATA)` (pamiętaj o nawiasach!). To określa, że dany element ma dopuszczać jedynie zawartość tekstową.
3. Wpisz `>`, aby zakończyć definicję elementu (rysunek 6.4).

Wskazówka

- ✓ Skrótowiec PCDATA (ang. *parsed character data*) oznacza parsowane dane znakowe i odnosi się do wartości tekstowej elementu. Te dane znakowe będą parsowane lub analizowane przez procesor XML.
- ✓ Tekst (zwany również ciągiem znaków) może być dowolną serią liter, liczb i symboli, np. „Halo”, „Privet Dr. 4” lub „99811”.
- ✓ Element, który został zdefiniowany tak, aby zawierać PCDATA, nie może zawierać żadnego innego elementu.
- ✓ W języku XML wielkość liter ma znaczenie. Słowo `<!ELEMENT` ma być zapisywane dokładnie w taki właśnie sposób. Zapis `<!Element` po prostu nie jest poprawny. Nie zapominaj także o wykrzykniku. Dla elementu możesz zastosować *nazwę* o zapisie mieszanym (małe i wielkie litery), o ile zawsze będziesz odwoływał się do niego dokładnie w ten sam sposób. Wiele aplikacji XML stosuje małe litery. Dzięki temu nie musisz poświęcać dodatkowego czasu na zapamiętywanie, jakiej wielkości znaków powinieneś użyć w każdym z przypadków.

Definiowanie pustego elementu

Poza elementami zawierającymi tekst, dokumenty DTD muszą mieć również możliwość definiowania **pustych elementów**. Jak wspomniano w rozdziale 1., pusty element nie ma żadnej własnej zawartości. Zamiast tego wykorzystuje on atrybuty służące do przechowywania danych (rysunek 6.5).

Aby zdefiniować pusty element:

1. Wpisz `<!ELEMENT znacznik`, gdzie *znacznik* jest nazwą elementu, który chcesz zdefiniować.
2. Następnie wpisz `EMPTY`, aby wskazać, że dany element nie będzie posiadał własnej zawartości tekstowej.
3. Wreszcie wpisz `>`, aby zakończyć definicję elementu (rysunek 6.6).

Wskazówki

- ✓ Zwróć uwagę, że wpisując `EMPTY`, nie używasz nawiasów, tak jak jest to wymagane przy (`#PCDATA`).
- ✓ Jak już wspomniano, puste elementy będą posiadały atrybuty, które zostały omówione w podrozdziale „Na temat atrybutów”.

xml
<pre>... <obrazek_glowny plik="lighthouse.jpg" szer="528" wys="349"/> <zrodlo id_sekcji="112" id_gazety="53"/> ...</pre>

Rysunek 6.5. W tym fragmencie kodu XML elementy `obrazek_glowny` i `zrodlo` są elementami pustymi. Nie ma znaczenia, czy wykorzystują one pojedynczy znacznik otwierający/zamykający, czy osobne znaczniki otwierające i zamykające. I tak oba są elementami pustymi

dtd
<pre><!ELEMENT obrazek_glowny EMPTY> <!ELEMENT zrodlo EMPTY></pre>

Rysunek 6.6. Oto fragment DTD definiujący elementy z rysunku 6.5. Definiowanie ich atrybutów zostało omówione w podrozdziale „Na temat atrybutów”

xml
<pre><?xml version="1.0"?> <cuda_starozytnosci> <cuda> <nazwa jezyk="polski">Kolos Rodyjski</nazwa> <nazwa jezyk ="grecki">Κολοσσός της Ρόδου</nazwa> <lokalizacja>Rodos, Grecja</lokalizacja> <wysokosc jednostka="metr">33</wysokosc> <historia> <rok wybudowania era="p.n.e.">282 </ rok_wybudowania> <rok_zniszczenia era="p.n.e.">226 </ rok_zniszczenia> <sposob_zniszczenia>trzęsienie ziemi </sposob_zniszczenia> <dzieje>W roku 294 p.n.e. mieszkańcy wyspy Rodos ...</dzieje> </historia> <obrazek_glowny plik="lighthouse.jpg" szer="528" wys="349"/> <zrodlo id_sekcji="112" id_gazety="53"/> </cuda> </cuda_starozytnosci></pre>

Rysunek 6.7. W tej uproszczonej wersji dokumentu głównego XML element `cuda_starozytnosci` zawiera pojedynczy element-dziecko o nazwie `cuda`

dtd
<pre><!ELEMENT cuda_starozytnosci (cuda)></pre>

Rysunek 6.8. Ta definicja DTD określa, że element `cuda_starozytnosci` może zawierać pojedynczy element-dziecko o nazwie `cuda`, tak jak pokazano na rysunku 6.7. Uwaga! Zawartość elementu `cuda` zależy tylko od jego definicji (a przynajmniej nie wpływa na nią definicja elementu `cuda_starozytnosci`)

Definiowanie elementu, który zawiera element-dziecko

Teraz, kiedy rozumiesz już, jak definiuje się podstawowe elementy XML w DTD, powinieneś nauczyć się definiowania elementów-rodziców, czyli elementów zawierających inne elementy (rysunek 6.7).

Aby zdefiniować element, który ma zawierać element-dziecko:

1. Wpisz `<!ELEMENT znacznik`, gdzie *znacznik* jest nazwą elementu, który chcesz zdefiniować.
2. Następnie wpisz (*dziecko*), gdzie *dziecko* jest nazwą elementu, który będzie zawarty w elemencie definiowanym.
3. Wreszcie wpisz `>`, aby zakończyć definicję elementu (rysunek 6.8).

Wskazówki

- ✓ Znacznik, który został zdefiniowany tak, aby zawierać jeden inny element, nie może zawierać niczego innego poza tym elementem. Nie może on zawierać innych dodatkowych elementów ani tekstu.
- ✓ Element-dziecko może być opcjonalny lub pojawiać się wielokrotnie. Więcej informacji na ten temat znajdziesz w podrozdziale „Definiowanie liczby wystąpień”.
- ✓ Możesz także kontrolować kolejność, w jakiej elementy powinny pojawiać się w dokumencie XML (patrz podrozdział „Definiowanie elementu, który zawiera kilka elementów-dzieci”).
- ✓ Jeśli zdefiniujesz, że dany element zawiera element-dziecko, to dany element musi zawierać wskazany element-dziecko w każdym wystąpieniu w dokumencie XML. Jeżeli tak nie będzie, dokument XML nie zostanie uznany za poprawny.

Definiowanie elementu, który zawiera kilka elementów-dzieci

Często dokument XML zawiera sekwencję elementów-dzieci (rysunek 6.9). W DTD możesz zdefiniować sekwencję elementów-dzieci, które muszą być zawarte w danym elemencie-rodzicu. Taka sekwencja definiuje również kolejność, w jakiej dane elementy-dzieci mają się pojawić.

Aby zdefiniować element zawierający elementy-dzieci:

1. Wpisz `<!ELEMENT znacznik`, gdzie `znacznik` jest nazwą elementu, który chcesz zdefiniować.
2. Następnie wpisz `(dziecko1`, gdzie `dziecko1` jest pierwszym elementem, który powinien pojawić się w danym elemencie-rodzicu.
3. Dalej wpisz `dziecko2`, gdzie `dziecko2` jest kolejnym elementem, który powinien pojawić się w danym elemencie-rodzicu. Elementy-dzieci oddzielaj przecinkiem i spacją.
4. Powtarzaj punkt 3. dla wszystkich elementów, które powinny pojawić się w elemencie-rodzicu.
5. Następnie wpisz `)`, aby zakończyć sekwencję.
6. Wpisz `>`, aby zakończyć definicję elementu (rysunek 6.10).

Wskazówki

- ✓ Najbardziej istotną rzeczą w sekwencji jest przecinek. Jest to znak rozdzielający elementy (lub grupy elementów) w sekwencji.
- ✓ Nie możesz używać (`#PCDATA`) w żadnej części sekwencji. Sekwencja musi zawierać jedynie elementy.
- ✓ Elementy zawarte w sekwencji mogą oczywiście zawierać własne elementy-dzieci. Element `historia` zdefiniowany na rysunku 6.10 zawiera faktycznie cztery indywidualne elementy-dzieci (co widać na rysunku 6.12).

```

xml
<?xml version="1.0"?>
<cuda_starozytnosci>
  <cud>
    <nazwa jezyk="polski">Kolos Rodyjski</nazwa>
    <lokalizacja>Rodos, Grecja</lokalizacja>
    <wysokosc jednostka="metr">33</wysokosc>
    <historia>
      <rok_wybudowania era="p.n.e.">282
    </rok_wybudowania>
      <rok_zniszczenia era="p.n.e.">226
    </rok_zniszczenia>
      <sposob_zniszczenia>trzęsienie ziemi
    </sposob_zniszczenia>
      <dzieje>W roku 294 p.n.e. mieszkańcy wyspy
        Rodos ...</dzieje>
    </historia>
    <obrazek_glowny plik="lighthouse.jpg"
      szer="528" wys="349"/>
    <zrodlo id_sekcji="112" id_gazety="53"/>
  </cud>
</cuda_starozytnosci>

```

Rysunek 6.9. Element `cud` posiada wiele elementów-dzieci. Zwróć uwagę, że spora część z tych elementów posiada atrybuty, które zostały omówione szczegółowo w podrozdziale „Na temat atrybutów”

```

dtd
<!ELEMENT cud (nazwa, lokalizacja, wysokosc,
historia, obrazek_glowny, zrodlo)>

```

Rysunek 6.10. Ta definicja DTD, wykorzystywana do walidacji dokumentu XML z rysunku 6.9, mówi, że element `cud` musi zawierać każdy z wymienionych elementów w danej kolejności. Nie może też zawierać niczego innego

dtd
<!ELEMENT cuda_starozytnosci (cud+)>
<!ELEMENT cud (nazwa+, lokalizacja, wysokosc, historia, obrazek_glowny, zrodlo*)>

Rysunek 6.11. Znaki specjalne (zwane również kwantyfikаторami) sprawiają, że dana definicja jest znacznie bardziej elastyczna. Teraz element `cuda_starozytnosci` musi zawierać co najmniej jeden (lub dowolnie więcej) element `cud`. Ponadto element `cud` musi zawierać co najmniej jeden (lub dowolnie więcej) element `nazwa`, a elementów `zrodlo` może być dowolna liczba (może ich też nie być w ogóle). Elementy `lokalizacja`, `wysokosc`, `historia` oraz `obrazek_glowny` muszą pojawić się dokładnie raz (co jest ustawieniem domyślnym)

dtd
<!ELEMENT historia (rok_wybudowania, rok_zniszczenia?, sposob_zniszczenia?, dzieje)>

Rysunek 6.12. Ta definicja elementu `historia` mówi, że element ten musi zawierać dokładnie jeden element `rok_wybudowania` i dokładnie jeden element `dzieje`. Elementy `rok_zniszczenia` i `sposob_zniszczenia` mogą być pominięte (lub mogą pojawić się co najwyżej raz)

Definiowanie liczby wystąpień

Jak dotąd element-rodzic mógł zawierać tylko jedną instancję każdego ze swoich elementów-dzieci. Jednak DTD dopuszcza więcej niż tylko jedną instancję elementu-dziecka w dokumencie XML. W DTD istnieją trzy specjalne symbole, które mogą być wykorzystane do zdefiniowania liczby wystąpień elementu-dziecka w elemencie-rodzicu (rysunki 6.11 i 6.12).

Aby zdefiniować liczbę wystąpień:

1. W definicji elementu, w części dotyczącej zawartości, wpisz nazwę elementu-dziecka.
2. Następnie wpisz *****, aby wskazać, że dany element-dziecko może pojawiać się w definiowanym elemencie-rodzicu tyle razy, ile to konieczne lub nie pojawiać się w ogóle (*zero lub kilka razy*).

Możesz też wpisać **+**, aby wskazać, że dany element-dziecko musi pojawić się w definiowanym elemencie-rodzicu dowolną liczbą razy, ale co najmniej raz (*jeden lub kilka razy*).

Możesz również wpisać **?**, aby wskazać, że dany element-dziecko może pojawić się w definiowanym elemencie-rodzicu nie więcej niż raz lub w ogóle (*zero lub jeden raz*).

Wskazówki

- ✓ Powtórzymy:
 - * oznacza zero lub kilka razy;
 - + oznacza jeden lub kilka razy;
 - ? oznacza zero lub tylko raz.
- ✓ Należy pamiętać, że element bez kwantyfikatora musi pojawić się dokładnie jeden raz.
- ✓ Kwantyfikatory możesz również używać do zdefiniowania liczby wystąpień dla sekwencji (patrz podrozdział „Definiowanie elementu, który zawiera kilka elementów-dzieci”).
- ✓ Nie istnieje żaden specjalny sposób definiowania konkretnej liczby występowania elementu (np. trzy wystąpienia). Jedyne (i dość obszerny) sposób to napisanie: (**e**lement, **e**lement, **e**lement).

Definiowanie wyborów

Czasem chcemy, żeby element XML zawierał jedną rzecz lub drugą — do wyboru (rysunek 6.13).

Aby zdefiniować wybory dla zawartości elementu:

1. Wpisz `<!ELEMENT znacznik`, gdzie *znacznik* jest nazwą elementu, który chcesz zdefiniować.
2. Następnie wpisz `(dziecko1`, gdzie *dziecko1* jest pierwszym elementem-dzieckiem, który może się pojawić.
3. Następnie wpisz `|`, aby wskazać, że jeśli pojawi się pierwszy element-dziecko, element zdefiniowany jako alternatywny już się pojawić nie może (i odwrotnie).
4. Dalej wpisz `dziecko2`, gdzie *dziecko2*, jest drugim elementem, który może się pojawić.
5. Powtórz czynności z punktów 3. – 4. dla każdego dodatkowego wyboru.
6. Wpisz `)`, aby zakończyć listę wyborów.
7. Wpisz `>`, aby zakończyć definicję elementu (rysunek 6.14).

Wskazówki

- ✓ Możesz dodać kwantyfikator `*` po punkcie 6. powyższej procedury, aby element mógł mieć dowolną liczbę dowolnych wyborów (patrz podrozdział „Definiowanie liczby wystąpień”).
- ✓ Kiedy kwantyfikator `*` zostanie zastosowany do listy wyborów, oznacza to, że dany element może zawierać dowolną liczbę indywidualnych wyborów w dowolnej kolejności. W efekcie powoduje to utworzenie nieuporządkowanej listy wyborów.
- ✓ W niektórych przypadkach możesz chcieć, aby element posiadał zarówno zawartość, jak i elementy-dzieci. Nazywa się to **zawartością mieszaną** (ang. *mixed content*) i zostało przedstawione na rysunku 6.13. W takiej sytuacji musisz dodać kwantyfikator `*` w sposób opisany w powyższych wskazówkach. **Uwaga!** Takiego rozwiązania nie używa się zbyt często podczas pisania własnego języka niestandardowych znaczników. Korzysta się z tego raczej w trakcie pisania dokumentu DTD, aby mógł on obsługiwać wiele dokumentów XML pochodzących z różnych źródeł.

```

xml
<cuda_starozytnosci>
  <cuda>
    <nazwa>Kolos Rodyjski</nazwa>
    <lokalizacja>Rodos, Grecja</lokalizacja>
  </cuda>
  <cuda>Piramida Cheopsa, Giza, Egipt</cuda>
  <cuda>Świątynia Artemidy w Efezie
    <miasto>Efez</miasto>
    <kraj>Turcja</kraj>
  </cuda>
</cuda_starozytnosci>

```

Rysunek 6.13. Wyobraź sobie, że informacje o cudach starożytności pochodzą z trzech różnych źródeł, a każde źródło ma inną strukturę elementu `cuda`

```

dtd
<!ELEMENT cuda_starozytnosci (cuda+)>
<!ELEMENT cuda (#PCDATA | nazwa | lokalizacja
| miasto | kraj)*>
<!ELEMENT nazwa (#PCDATA)>
<!ELEMENT lokalizacja (#PCDATA)>
<!ELEMENT miasto (#PCDATA)>
<!ELEMENT kraj (#PCDATA)>

```

Rysunek 6.14. To DTD wykorzystuje wybory, żeby obsłużyć różne struktury elementu `cuda` pokazane na rysunku 6.13. Definicja ta deklaruje, że element `cuda` może mieć zero lub więcej wystąpień będących typami PCDATA elementów `nazwa`, `lokalizacja`, `kraj` lub `miasto`

```

dtd
<!ELEMENT historia ((rok_wybudowania,
rok_zniszczenia, sposob_zniszczenia, dzieje)
| (rok_wybudowania, dzieje))>

```

Rysunek 6.15. Oto kolejny sposób na napisanie logiki DTD z rysunku 6.12. Tutaj DTD definiuje element `historia` w taki sposób, że może on zawierać albo elementy `rok_wybudowania`, `rok_zniszczenia`, `sposob_zniszczenia` i `dzieje`, albo jedynie elementy `rok_wybudowania` i `dzieje`

xml
<pre> <cuda_starozytnosci> <cuda> <nazwa>Kolos Rodyjski</nazwa> <lokalizacja>Rodos, Grecja</lokalizacja> </cuda> <cuda>Piramida Cheopsa, Giza, Egipt</cuda> <cuda>Świątynia Artemidy w Efezie <miasto>Efez</miasto> <kraj>Turcja</kraj> </cuda> <cuda> <nazwa>Mauzoleum w Halikarnasie</nazwa> <lokalizacja> <miasto>Bodrum</miasto> <kraj>Turcja</kraj> </lokalizacja> </cuda> </cuda_starozytnosci> </pre>

Rysunek 6.16. Tutaj do dokumentu XML z rysunku 6.13 dodałem kolejny element `cuda` (z nowego nieoczekiwane źródła). Ten nowy element `cuda` ma jeszcze inną strukturę. Zwróć uwagę, że w tym nowym elemencie element `lokalizacja` nie zawiera żadnych typów PCDATA. Jest on raczej elementem rodzicem dla elementów `kraj` i `miasto`

dtd
<pre> <!ELEMENT cuda_starozytnosci (cuda+)> <!ELEMENT cuda (#PCDATA nazwa lokalizacja miasto kraj)*> <!ELEMENT nazwa (#PCDATA)> <!ELEMENT lokalizacja ANY> <!ELEMENT miasto (#PCDATA)> <!ELEMENT kraj (#PCDATA)> </pre>

Rysunek 6.17. Zamiast do DTD z rysunku 6.14 dodawać definicję nowej struktury elementu `lokalizacja`, zdefiniowałem, aby ten element mógł zawierać cokolwiek. Nie jest to tak czytelne jak zdefiniowanie konkretnej zawartości dla elementu `lokalizacja`, ale działa

Definiowanie elementu, który ma dowolną zawartość

Chociaż DTD nie jest idealne do tworzenia uporządkowanego zestawu reguł, to w takim dokumencie możesz zdefiniować element, który ma dowolną zawartość. Oznacza to, że element ten może zawierać dowolną kombinację elementów i tekstu. Podobnie jak w przypadku zawartości mieszanej, jest to przydatne przy tworzeniu DTD, które ma obsługiwać dokumenty XML pochodzące z różnych źródeł. Dla niemożliwej do przewidzenia struktury elementów może to być jedyny sposób na zdefiniowanie elementów, które znasz i dopuszczasz (rysunek 6.16).

Aby zdefiniować element, który może mieć dowolną zawartość:

1. Wpisz `<!ELEMENT znacznik`, gdzie `znacznik` jest nazwą elementu, który chcesz zdefiniować.
2. Następnie wpisz `ANY`, aby dany element mógł zawierać dowolną kombinację elementów i parsowanych danych znakowych.
3. Wpisz `>`, aby zakończyć definicję elementu (rysunek 6.17).

Wskazówki

- ✓ Korzystając z instrukcji `ANY`, powinieneś zachować zdrowy rozsądek. Celem dokumentu DTD jest zdefiniowanie reguł dotyczących tego, co mogą, a czego nie mogą zawierać elementy. Jeśli chcesz dopuścić, aby każdy element miał dowolną zawartość, możesz równie dobrze w ogóle zrezygnować z DTD. Dokumenty DTD nie są wymagane. Pomagają po prostu zachować spójność danych.
- ✓ Instrukcja `ANY` definiuje, że dany element może zawierać dowolną strukturę. Jeśli jednak element ten zawiera elementy-dzieci, te elementy wciąż muszą być zdefiniowane w DTD. Innymi słowy — `ANY` nie pozwala, aby element zawierał elementy-dzieci, które nie zostały zdefiniowane w DTD. Wszystkie elementy pojawiające się w poprawnym dokumencie XML muszą być i tak zdefiniowane.
- ✓ Każdy element musi być zdefiniowany dokładnie raz (i tylko raz). Nawet jeśli jakiś element może pojawiać się w wielu różnych miejscach poprawnego dokumentu XML, nadal musi być zdefiniowany tylko raz.

Kilka słów o atrybutach

Atrybuty są użyteczne przy dostarczaniu dodatkowych danych *na temat* elementu. Informacje umieszczone w atrybutach raczej *opisują* zawartość dokumentu XML, a nie są samą zawartością.

Przykładowo jeśli w dokumencie głównym XML *Cuda starożytnego świata* element *nazwa* zawiera atrybut *język* *opisujący* język, w którym podana jest zawartość elementu *nazwa*, można by łatwo zrestrukturyzować dany dokument XML tak, aby ta sama informacja znajdowała się w dwóch indywidualnych elementach-dzieciach. Element *nazwa* mógłby zawierać dwa elementy: *język* oraz *nazwa_lokalna*.

Każdy sposób jest dobry, ale zasadniczo najlepsze praktyki sugerują, że elementy lepiej nadają się do przechowywania informacji, które mają być wyświetlane, a atrybuty lepiej sprawdzają się do przechowywania informacji o informacjach. Istnieje ku temu kilka przesłanek. Atrybuty nie mogą opisywać relacji między danymi, tak jak robią to elementy-dzieci, wartości atrybutów nie mogą być w prosty sposób zweryfikowane przez DTD i wreszcie atrybuty nie mogą zawierać kilku wartości, podczas gdy elementy-dzieci mają taką możliwość.

Atrybuty są oczywiście często wykorzystywane w pustych elementach, gdzie opisują informacje na temat danego elementu. Są one na przykład często stosowane do przechowywania identyfikatorów, ponieważ atrybuty nie są danymi, ale informacjami na temat tych danych (patrz podrozdział „Definiowanie atrybutów z wartościami unikatowymi”).

Wskazówka

- ✓ Decyzja co do sposobu, w jaki zdecydujesz się opracować swój język XML, powinna być oparta na jego zastosowaniu. Jeśli nie zamierzasz zbyt wiele „robić” z konkretną porcją informacji, wtedy utworzenie jej w formie atrybutu jest w porządku. Jeśli jednak zamierzasz korzystać z tej informacji w bardziej znaczący sposób, wtedy lepszym rozwiązaniem jest umieszczenie jej w postaci zawartości elementu (rysunek 6.18).

xml
<spособ_zniszczenia rok="426">pożar </spособ_zniszczenia>

xml
<rok_zniszczenia>426</rok_zniszczenia> <spособ_zniszczenia>pożar</spособ_zniszczenia>

Rysunek 6.18. *Oba fragmenty kodu XML zawierają tę samą informację: posąg Zeusa w Olimpii został zniszczony w roku 426 w wyniku pożaru. Różnica polega na tym, jak te informacje zostały zorganizowane. W pierwszym kodzie 426 jest wartością atrybutu. W drugim kodzie zarówno 426, jak i pożar stanowią zawartość indywidualnych elementów*

dtd
<!ELEMENT wysokosc (#PCDATA)>
<!ATTLIST wysokosc jednostka CDATA #IMPLIED>

Rysunek 6.19. Ta definicja atrybutu mówi, że element *wysokosc* może zawierać opcjonalny atrybut *jednostka* (z uwagi na status *#IMPLIED*), który zawiera tekst (z uwagi na typ atrybutu *CDATA*)

xml
<wysokosc>12</wysokosc>

xml
<wysokosc jednostka="metr">12</wysokosc>

xml
<wysokosc jednostka="12">metr</wysokosc>

Rysunek 6.20. Zgodnie z definicją DTD z rysunku 6.19, wszystkie widoczne na rysunku fragmenty kodu XML są prawidłowe, ponieważ atrybut *jednostka* jest opcjonalny (*#IMPLIED*), a jego zawartością może być dowolna kombinacja znaków

dtd
<!ELEMENT wysokosc (#PCDATA)>
<!ATTLIST wysokosc jednostka CDATA #REQUIRED>

Rysunek 6.21. Ta wersja definicji mówi, że w elemencie *wysokosc* atrybut *jednostka* jest wymagany

xml
<wysokosc>12</wysokosc>

xml
<wysokosc jednostka="metr">12</wysokosc>

xml
<wysokosc jednostka="12">metr</wysokosc>

Rysunek 6.22. Są to te same trzy przykłady, które zostały pokazane na rysunku 6.20. Jednak po przeprowadzeniu walidacji względem DTD z rysunku 6.21 tylko ostatnie dwa okazują się poprawne. Pierwszy fragment kodu nie jest prawidłowy, ponieważ element *wysokosc* nie zawiera atrybutu *jednostka*

Definiowanie atrybutów

Atrybut nie może pojawić się w poprawnym pliku XML dopóty, dopóki nie zostanie zadeklarowany w DTD.

Definicja atrybutu składa się z czterech części: nazwy elementu, nazwy atrybutu, typu atrybutu oraz statusu opcjonalnego.

Aby zdefiniować atrybut:

1. Wpisz **<!ATTLIST *znacznik***, gdzie *znacznik* jest nazwą elementu, w którym ma się pojawić dany atrybut.
2. Następnie wpisz *nazwa_atributu*, gdzie *nazwa_atributu* definiuje nazwę danego atrybutu.
3. Dalej wpisz **CDATA**, aby wskazać, że typem atrybutu jest tekst. W przeciwieństwie do danych typu (*#PCDATA*), *CDATA*, czyli *dane znakowe*, nie będą parsowane przez procesor.
4. Następnie dla opcjonalnego statusu wpisz **#IMPLIED**, aby wskazać, że w razie potrzeby dany atrybut może być pominięty (rysunek 6.19).

Możesz też wpisać **#REQUIRED**, aby wskazać, że dany atrybut nie może być pominięty i musi zawierać jakąś wartość (rysunek 6.21).

5. Wreszcie wpisz **>**, aby zakończyć definicję atrybutu.

Wskazówki

- ✓ Zwróć uwagę, że we wszystkich częściach definicji atrybutu ważna jest wielkość liter. Musisz je wpisywać w takiej samej formie, w jakiej podałem je w przykładach. Wpisanie na przykład *#Requird* nie ma w DTD żadnego znaczenia.
- ✓ Możesz zdefiniować wszystkie atrybuty dla danego elementu w pojedynczej definicji atrybutu. Przed zakończeniem definicji atrybutu w punkcie 5. powyższej procedury powtarzaj czynności z punktów 2. – 4. dla każdego atrybutu, który dany element powinien posiadać. Jest to najbardziej typowy sposób definiowania wielu atrybutów dla pojedynczego elementu.

Definiowanie wartości domyślnych

Zamiast wpisywania instrukcji #REQUIRED lub #IMPLIED w ramach opcjonalnego statusu atrybutu, możesz zdefiniować, aby atrybut ten posiadał wartości domyślne.

Aby zdefiniować atrybut z właściwościami domyślnymi:

1. Według punktów 1., 2. i 3. z procedury opisanej w podrozdziale „Definiowanie atrybutów” zdefiniuj nazwę elementu, nazwę atrybutu oraz typ atrybutu.
2. Następnie wpisz **"wartość_domyślna"** (cudzysłów otwierający i zamykający jest wymagany), gdzie **wartość_domyślna** będzie wartością tego atrybutu, jeśli żadna inna wartość nie zostanie zdefiniowana w dokumencie XML (rysunek 6.23).

Możesz też wpisać #FIXED **"wartość_domyślna"**, gdzie **wartość_domyślna** będzie wartością tego atrybutu, jeśli żadna wartość nie zostanie w wyraźny sposób zdefiniowana. *Ponadto* jeśli wartość tego atrybutu zostanie zdefiniowana, to musi ona być równa wartości domyślnej, aby dany dokument XML był poprawny (rysunek 6.25).

3. Wpisz >, aby zakończyć definicję atrybutu.

Wskazówki

- ✓ Jeśli definiujesz atrybut z wartością domyślną, parser XML automatycznie doda tę wartość domyślną, jeżeli atrybut nie zostanie ustawiony w danym dokumencie XML (rysunek 6.24).
- ✓ Jeśli definiujesz atrybut z instrukcją #FIXED **"wartość_domyślna"**, wartość atrybutu w dokumencie XML *musi* być ustawiona na **wartość_domyślną**, jeżeli jest w ogóle ustawiona. Jeśli atrybut nie jest ustawiony, parser automatycznie ustawi dla niego wartość domyślną (rysunek 6.26).
- ✓ Nie możesz łączyć wartości domyślnej z instrukcją #REQUIRED lub #IMPLIED. W rzeczywistości, skoro wartość domyślna jest już zdefiniowana, status opcjonalny praktycznie nie ma sensu.

dtd
<!ELEMENT wysokosc (#PCDATA)> <!ATTLIST wysokosc jednostka CDATA "metr">

Rysunek 6.23. Tym razem dodają domyślną wartość metr dla atrybutu wysokosc

dtd
<!ELEMENT wysokosc (#PCDATA)> <!ATTLIST wysokosc jednostka CDATA #FIXED "metr">

Rysunek 6.25. Wartość ustalona może być użyteczna, jeśli chcesz mieć pewność, że dany atrybut będzie miał określoną wartość, bez względu na to, czy faktycznie pojawi się w danym dokumencie XML

xml
<wysokosc jednostka="metr">39</wysokosc>

xml
<wysokosc jednostka="stopa">39</wysokosc>

xml
<wysokosc>39</wysokosc>

Rysunek 6.24. Wszystkie te fragmenty kodu XML są poprawne. Atrybut jednostka może posiadać dowolną wartość, a nawet może zostać pominięty. Jeśli — tak jak w trzecim kodzie — atrybut jednostka zostanie pominięty, parser zachowa się tak, jakby ten atrybut był w rzeczywistości obecny, a jego wartością była stopa

xml
<wysokosc jednostka="metr">39</wysokosc>

xml
<wysokosc jednostka="stopa">39</wysokosc>

xml
<wysokosc>39</wysokosc>

Rysunek 6.26. Są to te same trzy przykłady, które zostały pokazane na rysunku 6.24. Jednak po przeprowadzeniu walidacji względem definicji DTD pokazanej na rysunku 6.25, środkowy kod nie jest już poprawny. Jeśli dany atrybut zostanie ustalony, musi posiadać wartość metr (a nie stopa czy inny ciąg znaków). Zwróć uwagę, że w przypadku trzeciego fragmentu kodu parser zachowuje się tak, jakby wartością atrybutu jednostka faktycznie był metr

dtd
<pre><!ELEMENT wysokosc (#PCDATA)> <!ATTLIST wysokosc jednostka (metr centymetr) #REQUIRED></pre>

Rysunek 6.27. W tym przykładzie chcę dopuścić tylko dwie możliwe wartości dla atrybutu `jednostka` w elemencie `wysokosc`: `metr` lub `centymetr`. Lista wyborów umieszczona jest w nawiasach, a do rozdzielenia pozycji zastosowałem pionową kreskę. Zwróć uwagę, że dany atrybut musi być ustalony (z powodu wartości `#REQUIRED`)

xml
<pre><wysokosc jednostka="metr">39</wysokosc></pre>
xml
<pre><wysokosc jednostka="stopa">39</wysokosc></pre>
xml
<pre><wysokosc>39</wysokosc></pre>

Rysunek 6.28. Spośród tych trzech fragmentów kodu XML tylko pierwszy jest poprawny względem definicji DTD z rysunku 6.27. Środkowy kod jest niepoprawny, ponieważ `metr` nie jest dopuszczalnym wyborem dla zawartości tego atrybutu. Trzeci fragment jest niepoprawny, ponieważ brakuje atrybutu `jednostka`, chociaż jest on wymagany (`#REQUIRED`)

Definiowanie atrybutów z wyborami

Dokumenty DTD obsługują typy atrybutów, które dopuszczają znacznie więcej niż tylko dane znakowe. Jeden z takich typów pozwala definiować atrybut, który obsługuje różne predefiniowane wybory (rysunek 6.27).

Aby zdefiniować atrybut z wyborami:

1. Rozpocznij definicję atrybutu według punktów 1. i 2. procedury opisanej w podrozdziale „Definiowanie atrybutów”.
2. Wpisz (*wybór_1* | *wybór_2*), gdzie *wybór_n* reprezentuje każdą możliwą wartość atrybutu, a każdy atrybut w dokumencie XML może stosować tylko jeden z wymienionych wyborów. Wybory powinny być rozdzielone pionową kreską, a cała lista powinna być ujęta w nawiasy.
3. Określ opcjonalne statusy atrybutu w sposób opisany w punkcie 4. procedury z podrozdziału „Definiowanie atrybutów” oraz w punkcie 2. procedury z podrozdziału „Definiowanie wartości domyślnych”.
4. Wpisz `>`, aby zakończyć definicję atrybutu.

Wskazówki

- ✓ Przy definiowaniu każdego wyboru z listy należy przestrzegać reguł pisania poprawnych nazw XML (patrz rozdział 1., podrozdział „Tworzenie elementu głównego”).
- ✓ Istnieje jeszcze kilka innych rodzajów typów atrybutów: ID, IDREF i IDREFS (które zostały omówione w podrozdziałach „Definiowanie atrybutów z wartościami unikatowymi” i „Odwolywanie się do atrybutów z wartościami unikatowymi”), NMTOKEN i NMTOKENS (które zostały omówione w podrozdziale „Ograniczanie atrybutów do poprawnych nazw XML”) oraz ENTITY (opisany w rozdziale 7.).

Definiowanie atrybutów z wartościami unikatowymi

Istnieje kilka specjalnych rodzajów typów atrybutów. Atrybuty ID są definiowane z wartością, która jest unikatowa (niepowtarzalna) dla całego dokumentu XML. Atrybut ID jest idealny dla kluczy i innych informacji identyfikujących (kody produktów, kody klientów itd.).

Aby zdefiniować atrybuty ID:

1. Rozpocznij definicję atrybutu według punktów 1. i 2. procedury opisanej w podrozdziale „Definiowanie atrybutów”.
2. Wpisz **ID**, aby zdefiniować, że wartość tego atrybutu będzie unikatowa i niepowtarzalna dla całego dokumentu XML. Innymi słowy, żaden inny element nie może mieć atrybutu o tej samej wartości.
3. Określ opcjonalne statusy atrybutu w sposób opisany w punkcie 4. procedury z podrozdziału „Definiowanie atrybutów” (Uwaga! Dla atrybutów ID można stosować tylko instrukcję **#REQUIRED** lub **#IMPLIED**. Nie mogą one posiadać domyślnych wartości z punktu 2. procedury opisanej w podrozdziale „Definiowanie wartości domyślnych”).
4. Wpisz **>**, aby zakończyć definicję atrybutu (rysunek 6.29).

Wskazówki

- ✓ Dokument XML nie jest uznawany za poprawny, jeśli dwa elementy z atrybutami ID mają tę samą wartość — bez względu na to, czy nazwy elementów lub atrybutów są takie same, czy inne.
- ✓ Jedynym wyjątkiem od tej reguły jest możliwość zdefiniowania nieograniczonej liczby pomijanych atrybutów ID, z których każdy implikuje pustą wartość.
- ✓ Wartość atrybutu ID musi stosować się do tych samych reguł, które określają zasady pisania nazw XML (patrz rozdział 1., podrozdział „Tworzenie elementu głównego”). (Oznacza to, że atrybut ID nie może zawierać jedynie wartości liczbowych, jak to ma miejsce w przypadku wielu pól ID w bazach danych, w numerach ubezpieczenia itd., chyba że poprzedzisz je literą lub znakiem podkreślenia).

```

dtd
<!ELEMENT cud (nazwa)>
<!ATTLIST cud kod ID #REQUIRED>

```

Rysunek 6.29. Jeśli zamierzasz utworzyć atrybut ID, najlepiej, żeby był on wymagany. Jeśli nie, musi on być zasugerowany (**#IMPLIED**), ponieważ atrybuty ID nie mogą mieć wartości domyślnych

```

xml
<cud kod="w_143">
<nazwa jezyk="polski">Wiszące ogrody Semiramidy
</nazwa>
</cud>
<cud kod="w_284">
<nazwa jezyk="polski">Posąg Zeusa w Olimpii
</nazwa>
</cud>

```

```

xml
<cud kod="w_284">
<nazwa jezyk="polski">Wiszące ogrody Semiramidy
</nazwa>
</cud>
<cud kod="w_284">
<nazwa jezyk="polski">Posąg Zeusa w Olimpii
</nazwa>
</cud>

```

Rysunek 6.30. Zgodnie z tym, co zostało zdefiniowane w DTD z rysunku 6.29, atrybut kod musi za każdym razem zawierać unikatową wartość w całym dokumencie XML. Przy tym założeniu pierwszy fragment jest poprawny, ale drugi nie

dtd
<pre><!ELEMENT specjalna_strona (tytul, url)> <!ATTLIST specjalna_strona identyfikator_cudu IDREF #REQUIRED></pre>

Rysunek 6.31. Element `specjalna_strona` będzie dokumentował strony WWW poświęcone każdemu z cudów. Jego atrybut `identyfikator_cudu` został zdefiniowany jako typ `IDREF`, więc może zawierać identyfikator cudu, którego dotyczy

dtd
<pre><!ELEMENT ogolna_strona (tytul, url)> <!ATTLIST ogolna_strona zawartosc IDREFS #REQUIRED></pre>

Rysunek 6.33. Element `ogolna_strona` posiada atrybut `IDREFS` o nazwie `zawartosc`. Atrybut ten może zawierać listę identyfikatorów cudów, na których koncentruje się element `ogolna_strona`

xml
<pre><ogolna_strona zawartosc="w_143 w_284"> <tytul>Cuda starożytnego świata</tytul> <url>www.wonders_of_the_world.com</url> </ogolna_strona></pre>

Rysunek 6.34. Ten fragment kodu XML jest prawidłowy, jeśli zastosujemy do niego definicję DTD z rysunku 6.33

xml
<pre><specjalna_strona identyfikator_cudu="w_143"> <tytul>The Lost Gardens</tytul> <url>www.lost-gardens.com</url> </specjalna_strona> <specjalna_strona identyfikator_cudu="w_143"> <tytul>Herodot w Babilonie</tytul> <url>www.herodotus.com/babylon</url> </specjalna_strona> <specjalna_strona identyfikator_cudu="w_286"> <tytul>Zeus w Olimpii</tytul> <url>www.olympiaszeus.com</url> </specjalna_strona></pre>

Rysunek 6.32. Zgodnie z DTD z rysunku 6.31, atrybut `identyfikator_cudu` musi zawierać wartość pochodzącą z istniejącego atrybutu `ID` z danego dokumentu. (Uwaga! Ten fragment kodu XML pochodzi z pliku XML, który zawiera również pierwszy z fragmentów kodu z rysunku 6.30)

Odwoływanie się do atrybutów z wartościami unikatowymi

Atrybut, którego wartość jest identyczna z wartością dowolnego istniejącego atrybutu `ID` (omówionego w podrozdziale „Definiowanie atrybutów z wartościami unikatowymi”) w dokumencie XML, jest zwany atrybutem `IDREF` (rysunek 6.31).

Atrybut, którego wartość stanowi rozdzielona białymi znakami lista wartości istniejących atrybutów `ID`, jest zwany atrybutem `IDREFS` (litera „S” to oczywiście konsekwencja utworzenia w języku angielskim liczby mnogiej — rysunek 6.33).

Aby odwołać się do atrybutów z unikatowymi wartościami:

1. Rozpocznij definicję atrybutu według punktów 1. i 2. procedury opisanej w podrozdziale „Definiowanie atrybutów”.
2. Wpisz `IDREF`, aby zdefiniować atrybut, który posiada wartość odpowiadającą wartości dowolnego istniejącego atrybutu `ID` (takiego jak ten, który został zdefiniowany według instrukcji opisanych w podrozdziale „Definiowanie atrybutów z wartościami unikatowymi”). Możesz też wpisać `IDREFS` (z „s” na końcu) dla zdefiniowania atrybutu, który zawiera kilka rozdzielonych białymi znakami wartości odpowiadających wartościom istniejących atrybutów `ID`.
3. Określ opcjonalne statusy atrybutu w sposób opisany w punkcie 4. procedury z podrozdziału „Definiowanie atrybutów” oraz w punkcie 2. procedury z podrozdziału „Definiowanie wartości domyślnych”.
4. Wpisz `>`, aby zakończyć definicję atrybutu.

Wskazówki

- ✓ Zwróć uwagę, że może być kilka atrybutów `IDREF`, które odnoszą się do tego samego `ID` (rysunek 6.32). Jest to zupełnie poprawne. Tylko samo `ID` musi być unikatowe dla jednego elementu.
- ✓ Żaden czynnik nie wyklucza powtarzających się pozycji z atrybutu `IDREFS`. Konstrukcja taka jak `zawartosc="w_143 w_143 w_143"` jest zupełnie poprawna dla parsera, bez względu na to, czy właśnie o to Ci chodziło. Jeśli chcesz mieć większą kontrolę nad zawartością elementów i atrybutów, musisz zrezygnować z DTD na rzecz XML Schema (patrz część IV).

Ograniczanie atrybutów do poprawnych nazw XML

Dokumenty DTD nie pozwalają na wpisywanie dużej ilości danych, ale jest jedno ograniczenie, które możesz zastosować do atrybutów. Wartość atrybutu zdefiniowanego jako typ `NMTOKEN` musi być poprawną nazwą XML. Oznacza to, że dana wartość musi rozpoczynać się od litery lub znaku podkreślenia i powinna zawierać jedynie litery, liczby, znaki podkreślenia, myślniki i kropki.

Aby zapewnić, że wartości atrybutów będą zgodne z regułami pisania nazw XML:

1. Rozpocznij definicję atrybutu według punktów 1. i 2. procedury opisanej w podrozdziale „Definiowanie atrybutów”.
2. Wpisz `NMTOKEN`, jeśli chcesz, żeby wartość atrybutu była poprawną nazwą XML (patrz rozdział 1., podrozdział „Tworzenie elementu głównego”).
3. Możesz też wpisać `NMTOKENS`, jeśli chcesz, żeby wartość atrybutu była listą poprawnych nazw XML rozdzielonych białymi znakami.
4. Określ opcjonalne statusy atrybutu w sposób opisany w punkcie 4. procedury z podrozdziału „Definiowanie atrybutów” oraz w punkcie 2. procedury z podrozdziału „Definiowanie wartości domyślnych”.
5. Wpisz `>`, aby zakończyć definicję atrybutu (rysunek 6.35).

Wskazówki

- ✓ Atrybuty `NMTOKEN` nie mogą zawierać żadnych białych znaków, co może być dobrym powodem, żeby zastosować właśnie ten konkretny typ atrybutów.
- ✓ Jeśli chcesz, aby wartość atrybutu była nie tylko poprawną nazwą XML, ale żeby była również unikatowa dla całego dokumentu XML, zastosuj `ID` zamiast `NMTOKEN` (patrz podrozdział „Definiowanie atrybutów z wartościami unikatowymi”).

```

dtd
<!ELEMENT cud_w EMPTY>
<!ATTLIST cud_w slowo_kluczowe NMTOKEN #REQUIRED>

```

Rysunek 6.35. W tym dość sztucznym przykładzie potrzebuję dla każdego cudu pojedynczego słowa, które mógłbym wykorzystać jako podstawowe słowo kluczowe w specjalnej aplikacji online o nazwie Cudowna Wycieczka. Utworzyłem więc element `cud_w` z atrybutem `slowo_kluczowe`. Aby ograniczyć wartość atrybutu `slowo_kluczowe` do jednego słowa (bez białych znaków), mogę zdefiniować go jako typ `NMTOKEN`

```

xml
<cud>
<cud_w slowo_kluczowe="kolos"/>
<nazwa jezyk="polski">Kolos Rodyjski</nazwa>
...
<cud>
<cud_w slowo_kluczowe="piramida cheopsa"/>
<nazwa jezyk="polski">Piramida Cheopsa</nazwa>
...

```

```

xml
<cud>
<cud_w slowo_kluczowe="kolos"/>
<nazwa jezyk="polski">Kolos Rodyjski</nazwa>
...
<cud>
<cud_w slowo_kluczowe="piramida_cheopsa"/>
<nazwa jezyk="polski">Piramida Cheopsa</nazwa>
...

```

Rysunek 6.36. Tylko drugi fragment kodu z tego przykładu zostanie uznany za poprawny względem DTD z rysunku 6.35. W pierwszym fragmencie kodu atrybut `slowo_kluczowe` ma wartość „piramida cheopsa”, która jest zapisana z wykorzystaniem spacji, a spacje nie są dopuszczalne w atrybutach `NMTOKEN`

& 32, 110
 ' 32
 &apost; 110
 > 32, 68, 110
 < 32, 68, 110
 " 32, 110

A

absolute location path, 57
 adnotacja, 136
 Ajax, 15

- jako koncepcja, 244
- narzędzia, 259
- podstawy technologii, 240
- użycie, 241
- wykorzystywana technologia, 240
- zastosowanie, 242

 Altova XMLSpy, 265
 ancestor nodes, 56
 anonimowy typ niestandardowy, 144
 anotacja map, 253

- znacznik miejsca, 254

 ANSI, 270
 anyType, 156
 aplikacja biurowa

- przechowywanie danych w formacie XML, 255

 aplikacja komputerowa

- przechowywanie znaków, 269

 aplikacja XML, 21
 argument

- default, 174
- fixed, 174
- n, 140

 arkusz stylów XSLT

- inicjowanie, 40
- kolejność przetwarzania, 58
- odwołania, 38
- szablon, 39
- uproszczony, 203
- załączenie reguł XML Schema, 209

 ascending, 50
 ASCII, 270
 aspekt, 146

- enumeracji, 148
- maksimów, 147
- minimów, 147
- wzorca, 150
- xs:enumeration, 148
- xs:fractionDigits, 152
- xs:length, 149
- xs:list itemType, 153
- xs:maxExclusive, 146
- xs:maxInclusive, 146
- xs:maxLength, 149
- xs:minExclusive, 147
- xs:minInclusive, 147
- xs:minLength, 149
- xs:pattern, 150
- xs:totalDigits, 152
- xs:union memberTypes, 154

 Asynchronous JavaScript and XML, 15, 240
 atomic value, 213
 atrybut, 22, 24, 102

- a przestrzenie nazw, 186
- background, 85
- border, 85
- break-before, 87
- column-count, 88
- column-gap, 88
- default, 143
- definicja, 103
- definiowanie, 172
- dodawanie, 29
- elementFormDefault, 193
- elementu regionu, 88
- elementu-rodzica
 - dziedziczenie, 82
- ENTITY, 116
- exclude-result-prefixes, 206
- extent, 85
- fixed, 143

- atrybut
 - font-size, 82
 - form, 193
 - grupy
 - definiowanie, 175
 - odwołania, 176
 - height, 51
 - href, 51
 - ID, 106
 - instrukcje, 106
 - wartość, 106
 - IDREF, 107
 - IDREFS, 107
 - informacje o informacjach, 102
 - język, 102
 - liczby całkowite, 73
 - margin, 85
 - master-name, 85
 - master-reference, 85
 - match, 58
 - zwracanie węzłów, 65
 - maxOccurs, 161, 163, 169
 - wartości, 162, 163
 - method, 205
 - format XML, 86
 - minOccurs, 161, 169
 - wartości, 162
 - mixed, 166
 - nazwa, 26, 29
 - NMTOKEN, 108
 - ograniczanie do nazw XML, 108
 - predefiniowanie zawartości, 174
 - przechowywanie danych, 96
 - referencje do encji nieparsowanej, 116
 - scale-to-fit, 83
 - select, 208
 - stosowanie szablonów, 53
 - węzeł bieżący, 58
 - zwracanie węzłów, 65
 - space-after, 82
 - span, 88
 - src, 51
 - statusy opcjonalne, 103
 - target, 51
 - targetNamespace, 194
 - toc, 258
 - typ prosty, 139
 - version, 203, 246
 - wartość, 23, 24, 29
 - domyślna, 104
 - generowanie z dokumentu, 51
 - początkowa, 174
 - węzła
 - wybieranie, 62
 - width, 51
 - wskazywanie zawartości, 174
 - wyjściowy, 51
 - wykorzystanie, 102
 - wymaganie, 173
 - wymuszenie braku, 173
 - xmlns, 183
 - xsi:noNamespaceSchemaLocation, 135
 - z wartościami unikatowymi, 106
 - odwołania, 107
 - z wyborami, 105
 - zakres lokalny, 186
 - attribute axis, 62
 - autotekst, 32
- B**
- baza danych
 - natywna, 235
 - z obsługą XML-a, 235
 - białe znaki, 24
 - atrybuty NMTOKEN, 108
 - binding sequence, 218
 - blok, 82
 - podział kolumnowy, 88
 - zawartości strony, 82
 - nadawanie stylu, 82
 - błędy
 - duchy, 125
 - składniowe, 28, 30
 - body region, 81
 - boolean expressions, 63
- C**
- Castro Elizabeth, 43
 - CDATA
 - typ atrybutu, 103
 - Character Data, 33
 - character encoding, 269
 - child elements, 22
 - child nodes, 56
 - complex type, 132, 155
 - container.xml, 257
 - content model, 160
 - content type, 158
 - content.opf, 258
 - context node, 57
 - CSS, 82
 - current node, 57

- custom markup language, 21
- czas
 - Greenwich, 141
- czcionka
 - rozmiary, 82
- D**
- dane
 - data i czas, 140
 - nieparsowane, 114
 - typy, 132
 - określanie, 138
 - wejściowe
 - parsowanie z wykorzystaniem wyrażeń regularnych, 223
 - typu nie-XML, 223
 - znakowe, 103
- default, 143
- Definicja Schematu XML, 131
- definicja typu dokumentu, 93
- deklaracja
 - typu dokumentu, 123
 - XML, 22, 25
 - xsl:function, 207
- descendant nodes, 56
- descending, 50
- div, 70
- Document Type Definition, 14
- dodawanie, 70
 - wartości węzłów, 76
- dokument DTD, 94
 - a przestrzenie nazw, 197
 - atributy
 - ograniczenie, 108
 - encje
 - ogólne, 111
 - zewnętrzne, 112
 - obsługa wielu dokumentów XML, 100
 - tworzenie, 121
 - wady i zalety, 128
 - wewnętrzny
 - deklarowanie i tworzenie, 124
 - zewnętrzny
 - deklarowanie, 123
 - publiczny, 126, 127
 - tworzenie, 122
- dokument RSS, 246
- dokument XML
 - analiza zawartości, 37
 - atributy ID, 106
 - białe znaki, 24
 - deklaracja, 25
 - drzewo węzłów, 38
 - elementy
 - kwalifikowane, 191
 - proste i złożone, 132
 - encje, 32
 - korzystanie, 111
 - encje nieparsowane
 - wstawianie, 117
 - formatowanie, 37
 - dla celów wydruku, 37
 - hierarchiczna reprezentacja, 38
 - kodowanie znaków, 270
 - komentarz, 31
 - narzędzia, 21
 - podział danych na fragmenty, 28
 - poprawność, 23, 123
 - powiązany zbiór, 122
 - powiązanie z arkuszem stylów, 38
 - przekształcanie, 37
 - atributy wyjściowe, 51
 - za pomocą XSLT, 38
 - przestrzenie nazw, 189
 - przetwarzanie
 - wyodrębnianie fragmentów, 74
 - przykład, 12, 22
 - reguły DTD, 94
 - spójność, 93
 - sprawdzenie zgodności
 - narzędzia, 94
 - walidacja
 - względem DTD, 125
 - wartość atrybutu, 23
 - wewnętrzne i zewnętrzne DTD, 124
 - wiązanie z dokumentem XML Schema, 135
 - wielkość liter, 23
 - wybieranie zbioru węzłów, 63
 - zasada pisania, 23
 - zestawy znaków i encji, 269
 - znaki specjalne, 32
- dokument XML Schema, 133
 - dodawanie adnotacji, 136
 - przestrzenie nazw, 189, 195
 - i lokalizacji, 189
 - rozpoczynanie, 134
 - struktura, 196
 - wiązanie z dokumentem XML, 135
- dokument XQuery, 227
- źródło, 228

- dokument XSL-FO
 - akapity i nagłówki, 82
 - dodawanie obrazków, 83
 - struktura, 80
 - szablon główny, 86
 - tworzenie, 81
 - za pomocą XSLT, 86
 - dokument XSLT
 - łączenie z XSL-FO, 86
 - przekształcanie dokumentu XML, 38
 - dokumenty wynikowe
 - generowanie, 204, 205
 - drzewo węzłów, 38, 56
 - reprezentacja, 38
 - XML, 212
 - DTD, 14, 93
 - atrybuty
 - deklarowanie, 103
 - obsługujące wybory, 105
 - element
 - instancje, 99
 - o dowolnej zawartości, 101
 - pusty, 96
 - zawierający element-dziecko, 97
 - zawierający kilka elementów-dzieci, 98
 - zawierający tekst, 95
 - encje, 109
 - parametryczne, 118
 - liczba wystąpień
 - definiowanie, 99
 - referencje
 - do encji nieparsowanych, 116
 - publiczne, 272
 - sprawdzanie formatów, 94
 - wady i zalety, 128
 - walidacja i używanie, 121
 - wewnętrzne, 121
 - deklaracje, 124
 - wybory, 100
 - zewnętrzne, 121
 - dtd.uri, 123
 - dyrektywa
 - przetwarzania XML, 228
 - xml-stylesheet, 228
 - dzielenie, 70
 - z resztą, 70
- E**
- eBook, 257
 - struktura, 258
 - Ecma International, 256
 - EditX XML Editor, 265
 - edytory XML, 21, 264
 - element, 24
 - all, 162
 - ATTLIST, 103, 116
 - bez prefiksu, 185
 - binding, 252
 - Body, 250
 - category, 246
 - channel, 246
 - definiowanie, 101
 - PCDATA, 95
 - wyborów, 100
 - definitions, 251
 - description, 246
 - długość
 - dokładna, 149
 - maksymalna, 149
 - minimalna, 149
 - ograniczenie, 149
 - DOCTYPE, 123
 - domyślna przestrzeń nazw, 183
 - dziecko, 22
 - nazwa, 27
 - sekwencja, 98
 - tworzenie, 27
 - wartości stylu, 82
 - wcięcia, 28
 - ELEMENT, 95
 - enclosure, 246
 - ENTITY, 118
 - Envelope, 250
 - fo:block, 80, 81, 82
 - fo:external-graphic, 83
 - fo:flow, 80, 89
 - fo:layout-master-set, 80, 81, 84
 - fo:page-number, 87
 - fo:page-sequence, 80, 81, 85, 89
 - fo:region-after, 87
 - fo:region-before, 85
 - fo:region-body, 80, 81
 - fo:root, 80, 84
 - fo:simple-page-master, 80, 81, 89
 - fo:static-content, 85
 - globalny, 177
 - odwołanie, 168
 - główny, 22, 23
 - tworzenie, 26
 - Header, 250
 - hexBinary, 149
 - image, 246
 - informacje wyświetlane, 102

- item, 246
- kml, 254
- kwalifikowany, 191
- limit cyfr, 152
- link, 246
- literalny, 39, 42
- lokalny, 168, 177
 - dodawanie, 192
- łączenie w grupy, 170
- manifest, 258
- margin, 80
- media:content, 247
- message, 251
- metadata, 258
- navMap, 258
- nazwa, 26
- nierezydujący, 183
- NOTATION, 114
- outerBoundaryIs, 254
- page-height, 80
- page-width, 80
- Point, 254
- Polygon, 254
- PolyStyle, 254
- położenie, 26, 27
- portType, 251
- predefiniowanie zawartości, 143
- prefiks przestrzeni nazw, 184
 - oznaczanie, 185
- przydzielanie typu danych, 132
- pusty
 - definiowanie, 96
 - stosowanie, 30
 - typ złożony, 165
 - znaczniki, 23
- regionu, 80
- rodzic, 26
 - definiowanie, 97
- rss, 246
- sekwencja
 - definiowanie liczby wystąpień, 99
- service, 252
- spine, 258
- Style, 254
- styleURL, 254
- title, 246
- typ prosty, 132
 - definiowanie, 138
- typ złożony, 132, 155
 - rodzaje, 132
- types, 251
- wartość
 - adres URL, 117
- wyzolowanie zasięgu, 177
- wyjściowy
 - atrybuty i wartości, 51
- wykomentowana sekcja, 31
- wyświetlanie jako tekst, 33
- wzorzec
 - określanie, 150
- xs:all, 162
- xs:annotation, 136
- xs:attributeGroup, 175
- xs:choice, 163
- xs:complexContent, 157
- xs:complexType, 158, 159
- xs:documentation, 136
- xs:element, 138
- xs:extension, 164, 167
- xs:group, 170
- xs:import, 195
- xs:include, 194, 195
- xs:restriction, 164, 167
- xs:schema, 134
- xs:simpleContent, 157
- xsl:apply-templates, 52, 53, 86, 203
- xsl:attribute, 88
- xsl:choose, 48
- xsl:for-each, 46, 47
- xsl:for-each-group, 208
- xsl:import-schema, 209
- xsl:sort, 50, 219
- xsl:stylesheet, 86
- xsl:text, 60
- xsl:value-of, 44, 46
- z zawartością
 - mieszaną, 166
 - prostą, 156
 - złożoną, 156
- zagnieżdżanie, 23, 28
- zawierający
 - tekst, 95
 - wyłącznie tekst, 164
- złożony, 156
 - z typów prostych, 154
- znaczniki, 24
- em, 82
- encja.uri, 112, 115
- encje, 32, 109
- &, 272
- ', 272
- >, 272

encje

- <, 272
 - ", 272
 - nieparsowane, 114
 - definiowanie, 115
 - referencje, 116
 - wstawianie do dokumentu XML, 117
 - zawartość, 115
 - ogólne, 109
 - korzystanie, 111
 - tworzenie, 110
 - ogólne wewnętrzne, 110
 - ogólne zewnętrzne
 - korzystanie, 113
 - tworzenie, 112
 - parametryczne, 109
 - a grupy atrybutów, 176
 - a nazwana grupa modelowa, 170
 - tworzenie i wykorzystanie, 118
 - zewnętrzne, 119
 - parsowane, 114
 - predefiniowane, 32, 272
 - referencja nazwy, 110
 - referencje, 272
 - reprezentacja
 - tekstu, 110
 - znaków Unicode, 269
 - udostępnianie, 113
 - zawartość w pliku zewnętrznym, 112
 - zdefiniowane, 110
 - znormalizowana lista, 113
- enclosed expressions, 227
- enumeration, 148
- ePub, 257
 - narzędzia, 260
 - podspecyfikacje, 258
- etykieta, 138, 144
- eXtensible Markup Language, 11
- eXtensible Stylesheet Language, 14

F

- facets, 146
- false, 77
- Filmweb
 - przyznawanie gwiazdek, 242
- finalne drzewo wynikowe, 205
- Fitzgerald Michael, 151
- fixed, 143
- formalny identyfikator publiczny, 126

format

- czasu, 140
 - czasu trwania, 140
 - daty, 140, 141
 - dnia, 141
 - ePub, 257
 - miesiąca, 141
 - pakietu Office 2007, 256
 - pliku
 - dla dokumentów biurowych, 255
 - ZIP, 256
 - roku, 141
 - RSS, 245
 - ułamki sekund, 141
 - xs:gYearMonth, 141
- formatowanie
- liczb, 72
- FPI, 126
- funkcja
- |, 77
 - agregująca, 215
 - analize-string(), 223
 - avg(), 214
 - ceiling(), 73, 207
 - ciągów, 77
 - collection(), 235
 - contains(ciąg1, ciąg2), 77
 - count(), 71, 214, 215
 - current-date(), 221
 - current-dateTime(), 221
 - current-group(), 208
 - current-time(), 221
 - distinct-values(), 219
 - do modyfikacji ciągu znaków, 75
 - doc(), 228, 229
 - exists(), 218
 - floor(), 73
 - format-date(), 221
 - format-dateTime(), 221
 - format-number, 212
 - format-number(), 72
 - zaokrąglanie liczby, 73
 - formatowanie
 - ciągu znaków, 216
 - liczb, 70
 - format-time(), 221
 - id(id_ciągu), 77
 - last(), 69
 - logiczna, 77
 - lower-case(), 216
 - match(), 223

matching-substring(), 223
 max(), 215
 min(), 215
 name(), 77
 name(zbiór_węzłów), 77
 normalize-space(), 77
 normalize-space(ciąg1), 77
 not(wyrażenie), 77
 pobierania, 241
 position(), 69
 ProcessAjax(), 241
 replace(), 223
 RequestAjax(), 241
 reverse, 213
 round(), 73
 string-length(), 77
 string-length(ciąg1), 77
 substring(s,f,n), 74
 substring-after(), 74, 219
 substring-before(), 74
 sum(), 76, 214, 215
 tokenize(), 223
 translate(), 75, 216
 typograficzna XSL-FO, 88
 unparsed-text(), 223
 upper-case(), 216
 węzłów, 77
 wyznaczanie wartości minimalnej, 215
 zaokrąglenie liczb, 73

funkcje UDF

- opcjonalne parametry wejściowe, 207
- przestrzeń nazw, 206
- rekurencyjne, 207
- richter, 234
- tworzenie, 206
- w XQuery, 234
- wywoływanie, 207

G

general entities, 109
 geoznakowanie, 253
 ghost errors, 125
 Google Earth, 253
 Google Suggest, 15, 242
 grafika.uri, 83
 group value, 208
 grupa modelowa, 160

- nazwana
 - definiowanie, 170
 - odwołanie, 171

wyboru, 163
 xs:all, 162
 xs:sequence, 161

H

HTML, 11, 240

- dokument, 22
- element główny, 26
- uniwersalność, 15

 HyperText Markup Language, 11

I

identyfikator notacji

- NDATA, 115

 IEC, 270
 iFrame, 244
 in, 82
 informacje identyfikujące, 106
 instrukcja

- #FIXED, 104
- #IMPLIED, 103
- #REQUIRED, 103
- (#PCDATA), 95
- (dziecko), 97
- (dziecko1, 98
- *, 60
- //, 126
- +//, 126
- ANY, 101
 - elementy-dzieci, 101
- attributeFormDefault, 192
- declare function, 234
- dwie kropki, 61
- elementFormDefault, 192
- EMPTY, 96
- encoding, 270
- exclude-result-prefixes, 206
- form, 193
- group-by, 208
- href, 205
- IDREF, 107
- ISO//, 126
- method, 204
- nazwa_typu_niestandardowego, 145
- NMTOKEN, 108
- odwołanie do definicji typu dokumentu, 123
- pojedyncza kropka, 59
- PUBLIC, 127
- ref, 168, 171

instrukcja

- schemaLocation, 195
- select, 45
- standalone, 113, 117
- SYSTEM, 112, 123
- targetNamespace, 188
- test, 48
- type, 138
- use, 173
- value, 146
- wynikowa, 204
- xmlns, 191
- xmlns:prefiks, 184, 190, 191, 198, 206
- xmlns:xsi, 189
- xsi:schemaLocation, 189
- xsl:apply-templates, 58
- xsl:attribute, 51
- xsl:choose, 49
- xsl:for-each, 226
- xsl:for-each-group, 208
- xsl:function, 206
- xsl:if, 49
- xsl:matching-substring, 223
- xsl:non-matching-substring, 223
- xsl:output, 42, 204
- xsl:param, 206
- xsl:result-document, 204, 205
- xsl:sort, 208
- xsl:validation, 209
- xsl:version, 203

instrukcja przetwarzania, 23, 25

- xml-stylesheet, 38

instrukcje_notacji, 114

integer, 132

internal general entities, 110

ISO, 126

ISO-8859-1, 270

iTunes, 248

J

JavaScript, 240

jednolity

- identyfikator zasobu, 113
- wskaźnik zasobu, 113

język

- opisu usług internetowych, 251
- silnie typizowany, 212
- składu, 79
- ścieżek XML, 55

XML

- zasada opracowania, 102
- wyrażeń regularnych, 150
- znaczników niestandardowych, 21

K

Keyhole Markup Language, 253

klauzula

- for, 231, 233
- join, 231
- let, 231
 - złączenie, 231
- order by, 230
 - złączenie, 231
- return, 230, 231
- where, 230, 233

klucz_grupowania, 208

KML

- narzędzia, 260
- plik, 254
- podstawy, 253
- wykorzystanie, 253
- źródło, 253

kod

- językowy w standardzie ISO, 126
- ukrycie fragmentu, 31
- załączanie HTML i JavaScript, 33

kodowanie znaków, 269

- rodzaje, 270

kolumna

- zawartość strony, 88

komentarz, 31, 136

- w XPath 2.0, 222

- w XQuery, 227

komponent XML Schema, 190

- kwalifikowane, 192
- podział na pliki, 194
- przestrzeń nazw
 - domyślna, 190
 - odwołania, 190
 - z prefiksem, 190
- załączanie, 194

komunikacja

- asynchroniczna, 241
- typu serwer-serwer, 249

komunikat

- odpowiedzi, 250
- SOAP, 249, 250
 - protokół transportowy, 252
- żądania, 250

konstruktor
 elementów, 227
kontener, 57
kontrola
 liczby wystąpień, 169
 nad zawartością elementów i atrybutów, 107
 wyświetlania funkcji matematycznych, 70
 zawartości elementów, 151
konwencja typograficzna, 16
konwersja ciągu znaków, 216
książki elektroniczne, 257
kwantyfikikator, 99
kwerenda, 225
kwerendowanie
 danych źródłowych, 226

L

liczba wystąpień
 maksymalna, 169
 minimalna, 169
liczby
 formatowanie, 72
 ujemne, 72
 zaokrąglenie, 73
limit
 cyfr w liczbie, 152
Liquid XML Studio, 266
lista, 153
 encji
 znormalizowana, 113
 wyborów, 100, 105
literal elements, 39
litery
 zmiana wielkości, 75
location paths, 56

M

Markup Validation Service, 125
metadane, 24
metoda
 wynikowa, 42
 HTML, 42
mixed content, 100
mniejsze
 lub równe, 68
 niż, 68
mnożenie, 70
mod, 70

model danych, 201
 XPath 2.0, 212
model groups, 160
model zawartości, 160

N

nagłówki
 tworzenie struktury, 85
 zawartość strony, 85
namespace, 134, 181
NaN, 142
narzędzia
 Ajax, 259
 ePub, 260
 KML, 260
 ODF i OOXML, 260
 pisanie dokumentów XML, 21
 proponowanie słów kluczowych, 242
 RSS, 259
 SOAP i WSDL, 259
 sugerowania witryn, 15
nawiasy okrągłe, 151
nazwa
 kwalifikowana, 187
 rozszerzona, 187
 uniwersalna, 187
nazwa_typu_złożonego, 159
nazwany typ niestandardowy, 144
nie równa się, 68
nieparsowane dane znakowe, 33
niestandardowy język znaczników, 13
 w XML, 94
node tree, 38
notacja, 114
 zawartości nieparsowanej, 114
numeryczne odwołania znakowe, 269
 dziesiętne, 271
 szesnastkowe, 271
 używanie, 271

O

OASIS, 255
obiekt
 XMLHttpRequest, 240, 241
OCF, 258
odejmowanie, 70
ODF, 255
 a OOXML, 256
 narzędzia, 260
 wsparcie, 255

odstęp_po, 82
 OOXML, 255, 256
 narzędzia, 260
 wsparcie, 256
 opakowanie, 249
 OpenOffice.org, 255
 OpenXML Writer, 256
 operacje arytmetyczne, 70
 operator
 and, 68
 cast, 221
 matematyczny, 70
 OPF, 258
 or, 68
 to, 213
 OPS, 258
 oś
 atrybutu, 62
 rodzaje, 62
 ośmiobitowy format transformacji, 270
 output method, 42
 overall structure, 80
 oXygen XML Editor, 265

P

page content, 80
 page master, 81
 parameter entities, 109
 parametr
 inherit, 82
 n, 69
 parent element, 26
 parent node, 56
 Parsed Character Data, 33
 parsed entities, 114
 parser
 analiza XML, 125
 automatyczna wartość domyślna atrybutu, 104
 encje, 114
 encje nieparsowane, 117
 formalny identyfikator publiczny, 127
 komentarze, 136
 prefiksy, 197
 walidacja względem XML Schema, 135
 parsowane dane znakowe, 33
 pary nazwa-wartość, 29
 PCDATA, 33
 placemark, 254

plik
 JAR, 255
 mimetype, 257
 wyjściowy
 do wydruku, 80
 dodanie tekstu HTML, 42
 dokument XSL-FO, 86
 HTML, 42
 ogólna struktura, 80
 okładka, 89
 zawartość strony, 80
 wyłącznie tekstowy, 122
 wynikowy
 walidacja, 209
 zewnątrzny
 zawierający DTD, 122
 Podcast, 248
 podciąg
 wyodrębianie, 74
 podspecyfikacja
 OCF, 258
 OPF, 258
 OPS, 258
 podszablony, 39
 podwłaściwość, 84
 podziały stron
 wstawianie, 87
 populating the namespace, 188
 porównywanie wartości, 68
 powiązane źródła danych
 łączenie, 233
 predefined entities, 32
 predykat, 59, 63
 zastosowanie, 63
 prefiks
 atrybut, 186
 nazwa przestrzeni nazw, 184
 przestrzeni nazw XML Schema, 139
 w elemencie, 184
 xs, 134, 139
 procesor
 kompatybilność wersji XSLT, 202
 XQuery, 225
 XSL-FO, 79
 processing instructions, 25
 projekt Gutenberg, 257
 protokół uzyskiwania szybkiego dostępu
 do obiektów, 249
 przekształcanie, 37
 proces, 38

przestrzeń nazw, 134, 181
 blogChannel RSS, 248
 docelowa, 188
 dodawanie
 atrybutów lokalnych, 192
 elementów lokalnych, 192, 193
 dokumenty DTD, 197
 domyślna
 deklarowanie, 183
 uchyłanie, 183
 funkcje UDF, 206
 instancji XML Schema, 135
 iTunes Podcasting, 248
 język XQuery, 234
 komponenty zdefiniowane lokalnie, 192
 Media RSS, 247
 nazwa, 181
 deklarowanie prefiksu, 184
 łączenie z nazwą elementu, 187
 projektowanie, 182
 prefiksowana, 183
 relacja między XML Schema a XML, 189
 schemat schematów, 196
 SOAP, 250
 walidacja dokumentów XML, 191
 wpływ na atrybuty, 186
 WSDL, 251
 wypełnianie, 188
 zapobieganie powiązaniu elementu, 193
pt, 82
punkt kodu, 271

Q

QName, 187
quantified expression, 218

R

ramka
 styl wizualny, 83
Really Simple Syndication, 15, 245
referencje
 encji, 272
 znakowe, 111
regex, 150
rejestr
 publiczny, 252
relative location path, 56
restrykcja
 xs:sequence, 157
retrieving function, 241

return expression, 220
root element, 22
root template, 39
root type, 156
rozszerzalny
 język arkuszy stylów, 14
 język znaczników, 12
rozszerzenie
 .dtd, 94, 122
 .ent, 112
 fo, 81
 .kml, 253
 .ncx, 258
 .xquery, 227
 .xsd, 134, 194
 .xsl, 39
RSS, 15
 agregator kanałów, 245
 czytnik kanałów, 245
 kanał, 245
 moduły, 247
 blogChannel, 248
 Media, 247
 narzędzia, 259
 podstawy, 245
 przestrzeń nazw, 246
 rozszerzanie, 247
 schematy, 246
 źródło, 245

S

schemat, 93
schemat DTD, 94
schemat schematów, 196
 jako domyślna przestrzeń nazw, 196
schemat.uri, 189
sekcja
 CDATA, 33
sekwencja, 161, 211, 213
 cechy, 213
 elementów, 98
 instrukcja xsl:sort, 219
 kwantyfikacja warunków, 218
 usuwanie zduplikowanych pozycji, 219
wartość
 maksymalna, 215
 minimalna, 215
 uśrednianie, 214
wiązania, 218, 220, 230
zapętlanie, 220

shorthand properties, 84
 sibling nodes, 56
 Simple Object Access Protocol, 249
 simple type, 132
 skalowanie obrazka, 83
 proporcjonalne, 83
 skrót, 110
 //, 65
 bieżącego węzła, 59
 SOAP, 249
 funkcje, 249
 koperta, 249
 narzędzia, 259
 schemat komunikatu, 250
 wymiana komunikatów, 249
 sortowanie
 liczb, 50
 malejące, 50
 rosnące, 50
 sekwencji, 219
 tekstu, 50
 SQL, 226
 standard
 Ecma, 256
 IDPF, 258
 stopka
 numerowanie stron, 87
 string, 132
 string value, 44
 strona
 bloki zawartości, 82
 nadrzędna, 81
 nagłówkek, 85
 podziały, 87
 szablon
 definiowanie, 84
 dodawanie, 89
 zawartość, 89
 WWW, 11
 zawartość
 definiowanie, 84
 w kolumnach, 88
 struktura płaska, 213
 strumienie zdjęć, 245
 style sheets, 40
 Stylus Studio, 266
 subskrypcja, 245
 symbol specjalny, 111
 szablon XSLT, 38
 dodanie tekstu HTML, 43
 etykieta, 39

główny, 39
 dodanie tekstu HTML, 42
 tworzenie, 41
 wbudowany, 41
 instrukcje, 39
 reguły
 definiowanie, 42
 stosowanie, 52
 ręczne, 52
 tworzenie, 52
 wyrażenie, 55

Ś

ścieżka lokalizacji, 56, 212
 atrybut węzła, 62
 bezwzględna, 57
 tworzenie, 64
 bieżący węzeł, 59
 porównywanie dwóch wartości, 68
 względna, 56

T

tabela HTML
 tworzenie, 47
 tabela znaków Unicode, 273
 target namespace, 188
 tekst, 95
 template, 38
 test expression, 217
 testowanie
 operacje arytmetyczne, 70
 porównanie dwóch wartości, 68
 pozycji węzła, 69
 za pomocą wyrażen warunkowych, 232
 transformacje XSLT, 41
 transforming, 37
 true, 77
 typ
 anonimowy, 144
 czasu, 141
 daty i czasu, 140
 element pusty
 definiowanie, 165
 główny, 156
 liczbowy, 142
 limity maksymalne i minimalne, 147
 listy, 153
 anonimowy, 153
 nazwany, 153

łączony, 154
 anonimowy, 154
 nazwany, 154
 MIME, 115, 257
 nazwany, 144
 niestandardowy
 nazwany, 145
 pusty element, 156
 wyłącznie element, 156
 wyłącznie tekst, 156
 definiowanie, 164
 xs:anyURI, 138
 xs:boolean, 138
 xs:date, 138, 140
 xs:dateTime, 140
 xs:decimal, 138, 142
 xs:duration, 140
 xs:float, 142
 xs:gDay, 141
 xs:gMonth, 141
 xs:gMonthDay, 141
 xs:gYear, 141
 xs:int, 142
 xs:integer, 142
 xs:negativeInteger, 142
 xs:nonNegativeInteger, 142
 xs:nonPositiveInteger, 142
 xs:positiveInteger, 142
 xs:restriction, 144
 xs:simpleType, 144
 xs:string, 138
 xs:time, 138, 140
 zawartości, 158
 zawartość mieszana, 156
 definiowanie, 166
 złożony, 155
 anonimowy, 158
 atributy, 172
 domyślna derywacja, 157
 dowolna kolejność elementów-dzieci, 162
 nazwany, 158, 159
 podstawy, 156
 sekwencja elementów-dzieci, 161
 tylko element, 160
 wywodzenie, 157
 wywodzenie z innych typów złożonych, 167
 zawierający elementy-dzieci, 160

U

UDDI, 252
 biznesowe, 252
 UDF, 206
 Unicode, 269
 punkty kodu
 źródło, 271
 Uniform Resource Identifier, 113
 Uniform Resource Locator, 113
 Uniform Resource Name, 113
 union type, 154
 Universal Description, Discovery, and Integration, 252
 uniwersalna nazwa zasobu, 113
 unparsed, 33
 unparsed entities, 114
 URI, 113
 nazwy przestrzeni nazw, 182
 URL, 113, 182
 URN, 113
 User Defined Functions, 206
 usługa internetowa, 249
 oparta na protokole SOAP, 250
 rejestr, 252
 WSDL, 251
 UTC, 141
 UTF-16, 270
 UTF-8, 270

V

VBScript, 244

W

W3C, 13, 18
 funkcje XPath, 67
 rekomendacja dla XML-a, 25
 XML Schema, 133
 XSL-FO, 79
 walidacja
 dokumentu XML
 względem XML Schema, 191
 plików wynikowych XSLT, 209
 pliku wklejonego, 125
 pod podanym adresem URI, 125
 wielu dokumentów XML, 121
 załadowanego pliku, 125
 validator, 125, 135, 267

- wartość
 - auto, 84, 87
 - całkowita liczba cyfr, 152
 - ciągu, 44
 - column, 87
 - dashed, 83
 - domyślna, 143, 174
 - dotted, 83
 - double, 83
 - enumeracji, 148
 - even-page, 87
 - font-size, 82
 - groove, 83
 - grupy, 208
 - hidden, 83
 - indefinite, 84
 - inset, 83
 - liczba cyfr w części ułamkowej, 152
 - medium, 83
 - najniższa możliwa, 147
 - najwyższa możliwa, 146
 - niepodzielna, 213
 - none, 83
 - odd-page, 87
 - określanie
 - dopuszczalnego zakresu, 146
 - dopuszczalnego zbioru, 148
 - outset, 83
 - porównywanie, 68
 - qualified, 192
 - ridge, 83
 - solid, 83
 - sumowanie, 76
 - średnia, 76
 - thick, 83
 - thin, 83
 - unqualified, 192
 - ustalona, 143, 174
- warunek
 - kwantyfikacja, 218
 - select, 47
 - testowanie, 49, 217
 - wysokości, 68
 - xsl:if, 48
- Web Service, 249
- well-formed, 23
- wewnętrzne opcje ogólne, 110
- węzeł, 38
 - bieżący, 57
 - odwołania, 59
 - skrót, 59
 - ustalanie, 58
 - dokumentu, 56
 - główny, 39
 - identyfikacja, 39
 - jako dane wyjściowe, 39
 - kontekstowy, 57, 59
 - odwoływanie, 59
 - liczenie, 71
 - lokalizowanie, 56
 - potomny, 56
 - pozyskiwanie wartości, 57
 - przetwarzanie
 - partiami, 46
 - warunkowe, 48
 - przodek, 56
 - rodzeństwo, 56
 - rodzic, 56
 - skrót, 60
 - sortowanie
 - domyślne, 50
 - przed przetworzeniem, 50
 - sprawdzanie pozycji, 69
 - wybieranie
 - bez znajomości położenia, 65
 - dzieci, 60
 - podzbioru, 63
 - rodzeństwa, 61
 - rodzica, 61
 - warunkowe, 63
 - wszystkich potomków, 65
 - wyświetlenie zawartości, 44
 - zapętlanie, 46
 - zbiór
 - w wyrażeniu, 48
 - wynikowy, 57
 - znajdowanie, 69
 - zlokalizowany, 57
- white space, 24
- większe
 - lub równe, 68
 - niż, 68
- właściwość
 - border-style, 83
 - border-width, 83
 - break-after, 87
 - break-before, 87
 - content-height, 83
 - inherit, 87
 - margin, 84
 - onreadystatechange, 241
 - open, 241
 - page-height, 84
 - page-width, 84

- send, 241
- skrótowa, 84
- WMHelp XMLPad, 266
- wrapper, 249
- WSDL, 251
 - narzędzia, 259
 - schemat, 251
 - źródło, 252
- wskaźnik strefy czasowej, 141
- wybory, 100
 - atrybuty, 105
 - warunkowe, 49
- wyzolowany zasięg, 177
- wykomentowanie sekcji, 31, 222
- wyrażenie, 44
 - //nazwa_elementu, 65
 - choose-when-otherwise, 217
 - else(), 232
 - every, 218
 - FLWOR, 230, 233
 - for, 220
 - if-then-else, 217, 232
 - kwantyfikowane, 218
 - logiczne, 63
 - not, 68
 - regularne, 150
 - select, 45
 - składnia języka, 55
 - some, 218
 - ścieżkowe, 229
 - testowe, 217, 232
 - XQuery, 227
 - warunkowe, 232
 - zamknięte, 227
 - ewaluacja, 232
 - zwrotne, 220
- wzorzec
 - ###0.00, 72
 - #000.0#, 72
 - składnia języka, 55

X

- XDM, 212
- XHTML, 204
- XLS-FO, 14
- XML, 11, 12
 - a HTML, 11, 14, 15
 - analizowanie źródła, 38
 - deklaracja wersji, 25

- dokument
 - elementy główne, 22
 - źródłowy, 44
- lista wyborów, 105
- narzędzia, 259, 263
 - do konwersji i walidacji, 267
- rozszerzanie, 14
- specyfikacja, 21
- struktura, 23
- strukturalne definicje, 14
- w praktyce, 15
- zalety języka, 13
- zarządzanie informacją, 11
- zasoby, 267
 - specyfikacje, 267
- zastosowanie, 239
- znaczniki, 14
- XML applications, 21
- XML Copy Editor, 265
- XML declaration, 22
- XML Infoset, 212
- XML Path Language, 55
- XML Schema, 14, 131
 - atrybuty
 - grupy, 175
 - predefiniowanie zawartości, 174
 - definicje
 - lokalne i globalne, 177
 - w kilku plikach, 194
 - z różnych przestrzeni nazw, 195
 - hierarchia typów, 156
 - język regex, 150
 - komentarze, 136
 - komponent, 190
 - liczba wystąpień, 169
 - podział pliku, 195
 - predefiniowanie zawartości elementu, 143
 - przestrzeń nazw, 134, 188
 - odwołania do komponentów, 190
 - typ złożony, 155
 - typy proste, 137
 - data i czas, 140
 - dla liczb, 142
 - niestandardowe, 137, 144
 - wbudowane, 139
 - wersje, 133
 - wyrażenia regularne, 151
 - wzorce, 150
 - zakres dopuszczalnych wartości, 146
 - zalety, 131

- XMLmind XML Editor, 266
 - XMLwriter XML Editor, 266
 - XPath, 14, 39, 55
 - działania arytmetyczne, 72
 - funkcje, 67
 - łączenie, 76
 - osie
 - rodzaje, 62
 - podstawy, 212
 - porównanie wersji, 211, 212
 - rodzaje węzłów, 57
 - tłumaczenie ścieżek lokalizacji, 56
 - wersja 2.0, 211
 - formatowanie ciągów znaków, 216
 - funkcje daty i czasu, 221
 - język silnie typizowany, 212
 - komentarze, 222
 - model danych, 212
 - przetwarzanie danych wejściowych
 - nie-XML, 223
 - sekwencje, 213
 - zapętlanie, 220
 - testowanie warunków, 217
 - wyrażenia, 213
 - XQuery, 55, 201, 225
 - a XSLT 2.0, 226
 - dokument, 227
 - funkcje użytkownika, 234
 - i bazy danych, 235
 - komentarze, 227
 - łączenie powiązanych źródeł danych, 233
 - predykaty, 229
 - przestrzeń nazw, 234
 - wyrażenia
 - ścieżkowe, 229
 - warunkowe, 232
 - XQuery/XPath Data Model, 212
 - XSD, 131
 - xsd.uri, 135
 - XSDL, 131
 - XSL, 14
 - dla formatowania obiektów, 37
 - dla przekształceń, 37
 - XSL-FO, 37, 79
 - ogólna struktura, 80
 - zawartość strony, 80
 - xsl-region-after, 87
 - xsl-region-before, 85
 - XSLT, 14, 37
 - arkusz stylów, 38
 - i przestrzeń nazw, 198
 - konwersja XML na HTML, 39
 - plik wynikowy transformacji, 205
 - rozszerzanie, 202
 - walidacja pliku wynikowego, 209
 - wersja 2.0, 201
 - funkcje definiowane przez użytkownika, 206
 - grupowanie danych wynikowych, 208
 - kilka dokumentów wynikowych, 205
 - łatwość zastosowania, 203
 - walidacja plików wynikowych, 209
 - wsparcie dla dokumentów wynikowych
 - XHTML, 204
 - wsteczna kompatybilność, 202
- ## Y
- Yahoo! Finance, 243
 - Yahoo! Search
 - Media RSS, 247
- ## Z
- zagnieżdżona struktura
 - if-then-else, 217, 232
 - zaokrąglenie liczb, 73
 - zawartość mieszana, 100
 - zbiór wyborów
 - sekwencja, 163
 - tworzenie, 163
 - znacznik
 - główny, 26
 - otwierający, 27
 - td, 47
 - tr, 47
 - w XML, 12
 - wszystko-w-jednym, 23
 - XML a HTML, 13
 - xsl:output, 42
 - xsl:stylesheet, 40
 - xsl:template, 41
 - zamknięcie, 22
 - zamykający, 23, 24, 27
 - znak
 - !=, 68
 - #, 72
 - %, 118, 119
 - &, 111, 113
 - &#, 271
 - &#x, 271
 - *, 60, 64, 70, 99, 100
 - //, 65

?, 99
@, 62
|, 100
+, 70, 99
0, 72
c, 74
funta, 51
kropki, 72
metoda przechowywania, 269
myślnika, 140
poprawna reprezentacja, 269
specjalny, 99
 (xy)*, 150
 (xy)?, 150
 (xy)+, 150
 (xyz){2}, 151
 [0-9], 150
 [abc], 150
 \d, 150
 \D, 150

kropka, 150
\s, 150
\S, 150
to | tamto, 151
x*, 150
x?, 150
x{5,}, 151
x{5,8}, 151
x{5}, 151
x+, 150
sposób kodowania, 270
Unicode, 273
wieloznaczny, 61

Ż

źródło
 podcastowe, 248
 przykładów, 17



PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>



SZYBKI START

XML

WYDANIE II

XML to uniwersalny tekstowy format prezentacji danych. Jasne zasady tworzenia dokumentów XML oraz prostota ich wykorzystania sprawiły, że stał się standardem wymiany danych. Format XML można zastosować na setki sposobów: serializacja obiektów czy komunikacja z webserwisami to najprostsze z nich. Żeby stworzyć swój pierwszy dokument XML, nie potrzebujesz żadnych specjalistycznych narzędzi. Wystarczy notatnik, ta książka i możesz poznawać świat formatu XML!

W trakcie lektury zapoznasz się z zasadami tworzenia dokumentów XML. Nauczysz się zagnieżdżać elementy, dodawać atrybuty, stosować puste elementy oraz komentować tworzony dokument. W kolejnych rozdziałach poznasz prawdziwą potęgę formatu XML — transformacje za pomocą XSLT. Dzięki nim będziesz w stanie przekształcić dokument źródłowy w dowolnie skonstruowany dokument docelowy. Ponadto przekonasz się, do czego mogą być przydatne dokumenty DTD, które opisują format pliku XML. Na koniec będziesz miał okazję poznać najnowsze zalecenia W3C w zakresie XML oraz wiele praktycznych zastosowań tego formatu. Ta książka jest kompendium wiedzy na temat najistotniejszych zagadnień związanych z formatem XML. Warto mieć ją zawsze pod ręką!

Dzięki tej książce:

- poznasz zasady tworzenia dokumentów XML
- wykorzystasz XSLT do transformacji XML
- sprawdzisz poprawność dokumentu dzięki DTD
- poznasz praktyczne zastosowania formatu XML

Wykorzystaj możliwości formatu XML!

Nr katalogowy: 17256



Księgarnia internetowa
<http://helion.pl>



Zamówienia telefoniczne:
0 801 339900



0 601 339900



Helion

Sprawdź najnowsze promocje:

• <http://helion.pl/promocje>

Książki najchętniej czytane:

• <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

• <http://helion.pl/nowości>

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel.: 32 230 98 63

e-mail: helion@helion.pl

<http://helion.pl>

helion.pl
księgarnia
internetowa

Cena 49,00 zł

ISBN 978-83-246-8237-9



9 788324 682379

Informatyka w najlepszym wydaniu