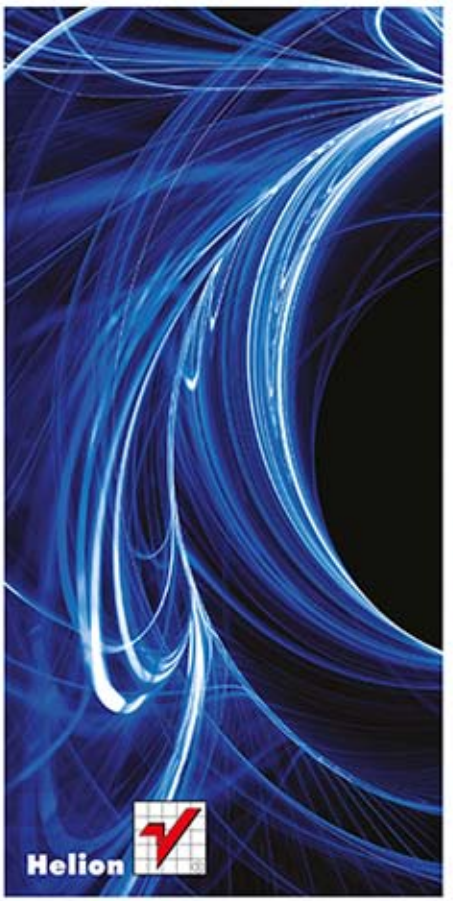
SZYBKI START

XML

WYDANIE II



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.

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.

Spis treści

Tworzenie dokumentów DTD 6

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 CśśML, (fikcyjny) język znaczników cudów starożytnego świata, będący systemem katalogowania danych dotyczących cudów starożytności. CśśML 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 *zweryfikować*, 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

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

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

```
<!ELEMENT cud  (nazwa, lokalizacja, wysokosc)>
<!ELEMENT nazwa (#PCDATA)>
<!ELEMENT lokalizacja (#PCDATA)>
<!ELEMENT wysokosc (#PCDATA)>
```

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>
 <cud>
  <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>
 </cud>
</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 posiadałby 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`, `wojewodztwo` 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

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

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

```
<!ELEMENT obrazek_glowny EMPTY>
<!ELEMENT zrodlo EMPTY>
```

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

```xml
<?xml version="1.0"?>
<cuda_starozytnosci>
 <cud>
  <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"/>
 </cud>
</cuda_starozytnosci>
```

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

```dtd
<!ELEMENT cuda_starozytnosci (cud)>
```

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

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 kwantyfikatorami) 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 liczbę 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

- ✔ Powtórzmy:

  * 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.
- ✔ Kwantyfikatorów 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). Jedyny (i dość obszerny) sposób to napisanie: **`(element, element, element)`**.

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

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

dtd

```
<!ELEMENT cuda_starozytnosci (cud+)>
<!ELEMENT cud (#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 cud pokazane na rysunku 6.13. Definicja ta deklaruje, że element cud 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
<cuda_starozytnosci>
 <cud>
  <nazwa>Kolos Rodyjski</nazwa>
  <lokalizacja>Rodos, Grecja</lokalizacja>
 </cud>
 <cud>Piramida Cheopsa, Giza, Egipt</cud>
 <cud>Świątynia Artemidy w Efezie
  <miasto>Efez</miasto>
  <kraj>Turcja</kraj>
 </cud>
 <cud>
 <nazwa>Mauzoleum w Halikarnasie</nazwa>
  <lokalizacja>
   <miasto>Bodrum</miasto>
   <kraj>Turcja</kraj>
  </lokalizacja>
 </cud>
</cuda_starozytnosci>
```

Rysunek 6.16. *Tutaj do dokumentu XML z rysunku 6.13 dodałem kolejny element cud (z nowego nieoczekiwanego źródła). Ten nowy element cud 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
<!ELEMENT cuda_starozytnosci (cud+)>
<!ELEMENT cud (#PCDATA | nazwa | lokalizacja
 | miasto | kraj)*>
<!ELEMENT nazwa (#PCDATA)>
<!ELEMENT lokalizacja ANY>
<!ELEMENT miasto (#PCDATA)>
<!ELEMENT kraj (#PCDATA)>
```

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 `jezyk` *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: `jezyk` 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

```
<sposob_zniszczenia rok="426">pożar
 </sposob_zniszczenia>
```

xml

```
<rok_zniszczenia>426</rok_zniszczenia>
<sposob_zniszczenia>pożar</sposob_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_atrybutu`***, gdzie *`nazwa_atrybutu`* 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 `#Required` 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_domyślnq*, 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 dodaję 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
<!ELEMENT wysokosc (#PCDATA)>
<!ATTLIST wysokosc jednostka (metr |
 centymetr) #REQUIRED>
```

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

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

```
xml
<wysokosc>39</wysokosc>
```

Rysunek 6.28. *Spośród tych trzech fragmentów kodu XML tylko pierwszy jest poprawny względem definicji DTD z rysunku 6.27. Środkowy kod jest nieprawidłowy, ponieważ metr nie jest dopuszczalnym wyborem dla zawartości tego atrybutu. Trzeci fragment jest nieprawidłowy, 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 „Odwoływanie 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
<!ELEMENT specjalna_strona (tytul, url)>
<!ATTLIST specjalna_strona identyfikator_cudu
 IDREF #REQUIRED>
```

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
<!ELEMENT ogolna_strona (tytul, url)>
<!ATTLIST ogolna_strona zawartosc IDREFS
 #REQUIRED>
```

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

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

```
xml
<specjalna_strona identyfikator_cudu="w_143">
<tytuł>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>
```

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 *`zawartość`*`="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*

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

Ś

T

U

V

W

X

Y

Z

Ź

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/nowosci

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