Obsługa XML w Javie

Ćwiczenia

Document Object Model

1. Przeglądanie całego dokumentu

1. Napisz program, który wczytuje dowolny dokument XML i korzystając z DOM (głównie interfejsu `Node`), rekurencyjnie przechodząc po drzewie dokumentu zbiera, a następnie wypisuje następujące informacje:
   a) liczba elementów o poszczególnych nazwach,
   b) sumaryczny rozmiar węzłów tekstowych,
   c) maksymalna głębokość drzewa dokumentu.
2. Niech program zbiera także informacje o liczbie atrybutów o poszczególnych nazwach.

2. Znajdowanie interesujących danych – wersja bez XPath

Napisz program, który wczytuje dokument sklep.xml (oczywiście nazwa konkretnego pliku ma być parametrem wiersza poleceń) i korzystając z DOM wyszukuje i wypisuje następujące informacje:

a) kategorie towarów,
b) dla każdej kategorii towar o najniższej cenie w tej kategorii.

Wykorzystaj interfejs `Element` i metody takie jak `getDocumentElement`, `getElementsByTagName`, `getAttribute, getTextContent`.

3. Obsługa przestrzeni nazw

Zmodyfikuj rozwiązanie zadania 2 tak, aby obsługiwało przestrzenie nazw i działało z dokumentami takimi jak sklep_ns.xml.

4. Modyfikacja i zapis dokumentu

Napisz program, który:

1. Wczytuje dokument (taki jak) sklep_ns.xml.
2. Do każdego towaru dodaje element `cena-brutto` z wyliczoną ceną brutto.
3. Zapisuje zmieniony dokument do pliku podanego w drugim parametrze programu.

Wsparcie dla dodatkowych standardów

5. Znajdowanie interesujących danych – wersja z XPath

1. Zrealizuj funkcjonalność zadania 2 wykorzystując wsparcie dla XPath.
2. (Opcja) Podobnie dla wersji z przestrzeniami nazw (zadanie 3). Zapewne trzeba zdefiniować odpowiedni namespace context.

6. Transformacje na wejściu i na wyjściu

1. Do programu z zadania 2 dodaj wykonanie przekształcenia XSLT na wczytywanym dokumencie, przed jego przetworzeniem jako drzewa DOM. Zastosuj arkusz sklep_filtr.xsl z parametrem `id-kategorii` wziętym z drugiego argumentu programu.
2. Do programu z zadania 4 zastosuj arkusz sklep-html.xsl i zapisz wynik (HTML) w pliku.

7. Walidacja przy odczycie

1. Do programów utworzonych w zadaniach 1-5 (lub tylko wybranych) dodaj walidację względem XML Schema podczas wczytywania dokumentu.
2. Sprawdź działanie programu na niepoprawnym pliku wejściowym (błąd składniowy, błąd w strukturze).

8. Walidacja drzewa DOM

1. Do programu z zadania 4 dodaj walidację drzewa DOM po przetworzeniu przez logikę programu, a przed zapisaniem wyniku do pliku.
   a) początkowo walidacja względem sklep.xsd powinna wskazywać na błędy (element `cena-brutto`),
   b) zmień schemat lub stwórz nowy, tak aby walidacja kończyła się sukcesem.

JAXB

9. Kompilacja schematu do klas Javy

1. Używając polecenia xjc w wierszu poleceń skompiluj schemat sklep.xsd do klas Javy. Obejrzyj wynikowe klasy.
2. To samo w wersji z przestrzeniami nazw (sklep_ns.xsd). Dalej w zadaniach z JAXB korzystamy z tej wersji.

10. Odczyt danych

Korzystając z mechanizmów JAXB oraz wygenerowanych wcześniej klas napisz program, który:

1. Wczytuje plik (taki jak sklep_ns.xml) podany w pierwszym parametrze programu i dla kategorii, której id podane jest jako drugi parametr programu, znajduje najtańszy produkt. Wypisywana jest nazwa produktu oraz cena.
2. (Opcja) W przypadku podania trzeciego parametru – daty w formacie YYYY-MM-DD, w wyszukiwaniu najtańszego produktu uwzględnia także ceny promocyjne obowiązujące danego dnia.

11. Modyfikacja i zapis dokumentu

Napisz program o 4 argumentach (nazwijmy je *A B C D*), który:

1. Wczytuje dokument taki jak sklep_ns.xml z pliku *A*.
2. Dla wszystkich towarów z kategorii podanej w parametrze C stosuje podwyżkę w wysokości *D* procent.
3. Zapisuje zmieniony dokument w pliku *B*.

JAXB – Dostosowywanie mapowania

12. Adnotacje w schemacie

Za pomocą adnotacji w schemacie (element `xs:appinfo`) wpłyń na generowany kod Javy:

1. Dodaj do pakietu i wybranej wynikowej klasy dokumentację Javadoc.
2. Zmień nazwy pól w Javie odpowiadających cenie netto i VAT.
3. (Opcja) Spraw, aby VAT był reprezentowany po stronie Javy jako typ prosty.
4. (Opcja) Spraw, aby cena netto po stronie Javy była reprezentowana jako pole typu `int` (lub `Integer`), w którym cena podana jest w groszach. Po stronie Javy należy zapewnić metody do parsowania i serializacji wartości z uwzględnieniem zapisu z dwoma miejscami po przecinku po stronie XML.

13. Adnotacje poza schematem

Zapisz adnotacje z zadania 12 w osobnym pliku zamiast w samym schemacie.

14. Wykorzystanie JAXB w scenariuszu Java → XML, adnotacje JAXB w Javie

1. Do dostarczonych klas modelu aplikacji „bankowej” dodaj odpowiednie adnotacje, aby możliwe było zapisanie obiektu klasy `Bank` do XML.
2. Poprzez dodatkowe adnotacje stopniowo doprowadź strukturę wynikowego dokumentu XML do zgodnej z przykładowym plikiem `bank_orig.xml`.
3. Używając narzędzia `schemagen` wygeneruj XML Schema dla stworzonej konfiguracji.
4. (Opcja) Przetestuj czy aplikacja poprawnie wczytuje plik `bank_orig.xml`.
5. (Opcja) Dodaj logikę zmiany stanu podanego konta o podaną wartość i zapisania zmienionego pliku.

SAX

15. Przeglądanie całego dokumentu

Używając standardu SAX, zrealizuj program o funkcjonalności analogicznej do programu z zadania 1, tj. czytający dowolny plik XML i wypisujący statystyki takie jak:

a) liczba elementów o poszczególnych nazwach,

b) sumaryczny rozmiar węzłów tekstowych,

c) maksymalna głębokość drzewa dokumentu.

16. Wyszukiwanie interesujących danych

Używając SAX-a napisz program, który

1. Wczytuje plik taki jak sklep.xml podany jako pierwszy argument.
2. Oblicza średnią cenę wszystkich towarów.
3. (Opcja) Wersja z obsługą przestrzeni nazw i pliku sklep_ns.xml.
4. (Opcja) Dodaj walidację względem XML Schema i wykorzystaj własną implementację interfejsu `ErrorHandler` do obsługi błędów.

17. (Opcja) Filtry SAX

1. Napisz filtr SAX, który nie przepuszcza zdarzeń dotyczących kategorii innych niż podana i towarów z tych kategorii.
2. Użyj tego filtra do realizacji programu, który działa jak program z zadania 16, ale uwzględnia tylko towary z podanej kategorii.

18. (Opcja) Transformacja jako sposób zapisania strumienia zdarzeń SAX

1. Napisz program, który łączy w sobie Transformer oraz filtr SAX z zadania 17 i zapisuje w wyniku plik analogiczny do sklep.xml, ale zawierający tylko towary z wybranej kategorii.
2. Wykorzystując `ValidatorHandler`, dodaj walidację „w locie” wyniku przed zapisaniem.

StAX

19. Przeglądanie całego dokumentu

Używając interfejsu `XMLStreamReader`, zrealizuj program o funkcjonalności analogicznej do programu z zadania 1, tj. czytający dowolny plik XML i wypisujący statystyki takie jak:

a) liczba elementów o poszczególnych nazwach,

b) sumaryczny rozmiar węzłów tekstowych,

c) maksymalna głębokość drzewa dokumentu.

20. Znajdowanie interesujących danych

Używając interfejsu `XMLEventReader` napisz program, który

1. Wczytuje plik taki jak sklep.xml podany jako pierwszy argument.
2. Oblicza średnią cenę wszystkich towarów.
3. (Opcja) Wersja z obsługą przestrzeni nazw i pliku sklep_ns.xml.

21. (Opcja) Filtry StAX

Jeśli podano drugi argument, powyższy program ma uwzględniać tylko towary z podanej kategorii. Użyj do tego filtrów StAX.

22. (Opcja) Zapisywanie w StAX

Napisz program o 4 argumentach (nazwijmy je *A B C D*), który:

1. Wczytuje dokument taki jak sklep.xml z pliku *A*.
2. Dla wszystkich towarów z kategorii podanej w parametrze C stosuje podwyżkę w wysokości *D* procent.
3. Zapisuje zmieniony dokument w pliku *B*.

Użyj do tego m.in. interfejsu `XMLEventWriter`.