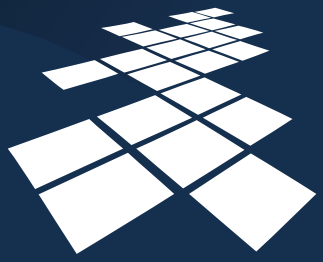
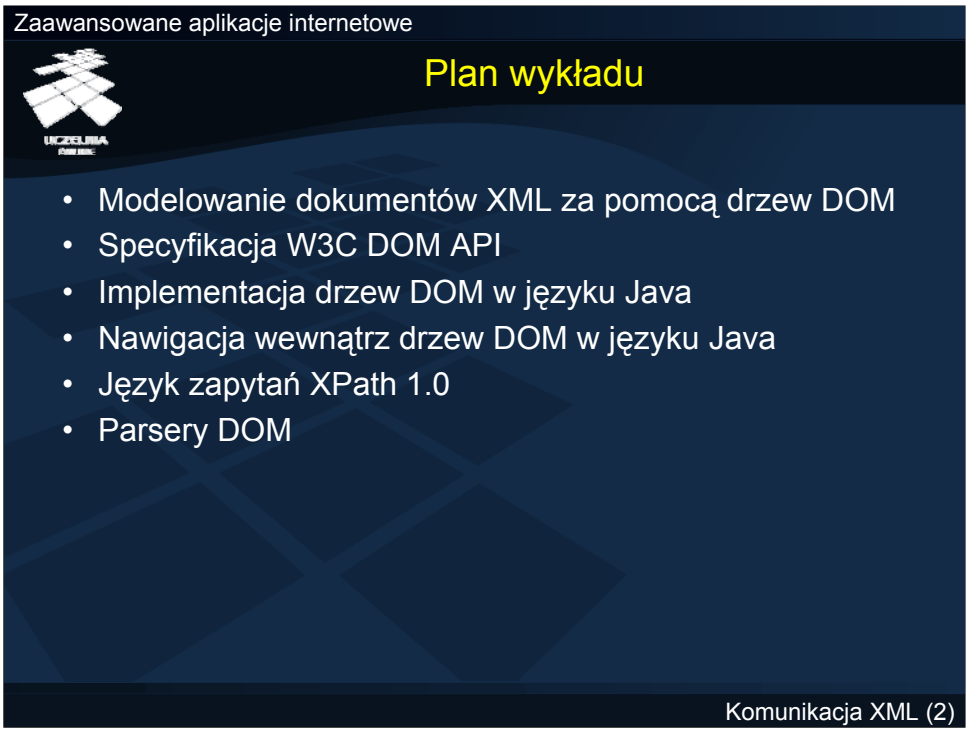
Wykład 2

Komunikacja XML

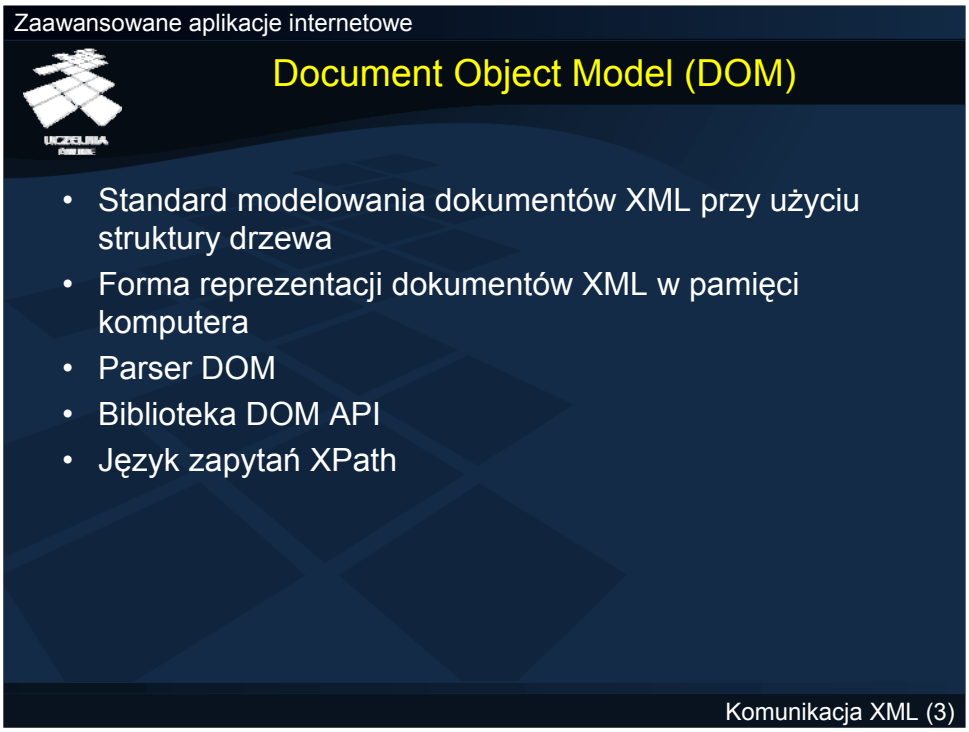
wykład prowadzi: Maciej Zakrzewicz



UCZELNIA
ONLINE

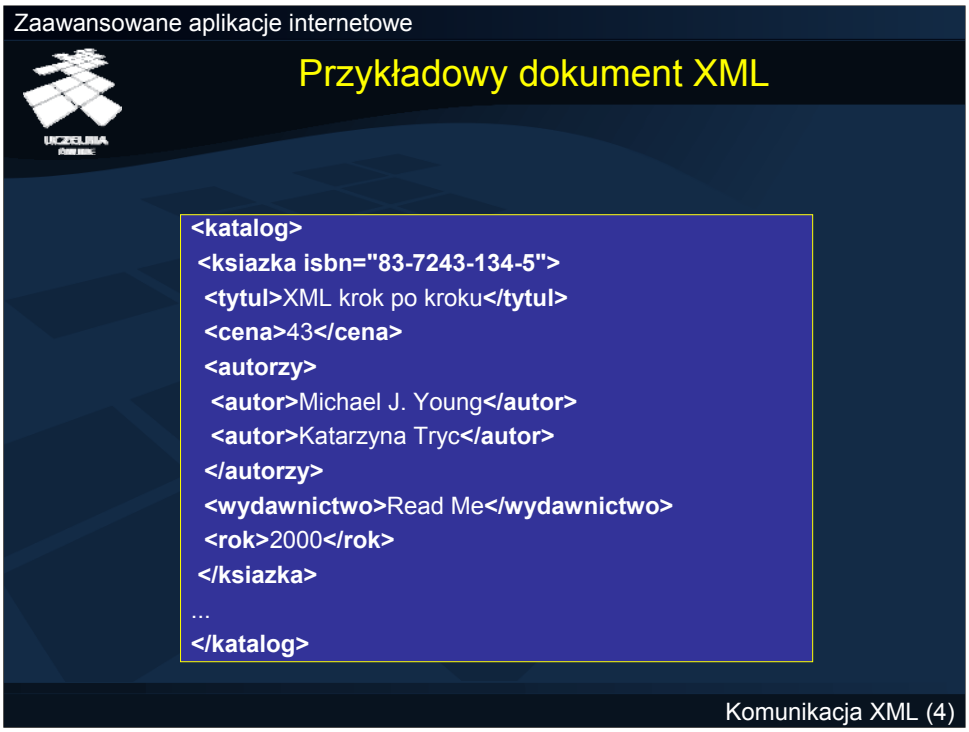


Celem wykładu jest omówienie podstawowych rozwiązań technologicznych umożliwiających implementację elektronicznej wymiany danych za pomocą dokumentów XML. Wykład obejmie: koncepcję drzew DOM i ich odniesienie do struktury dokumentów XML, specyfikację biblioteki programisty W3C DOM API, implementację drzew DOM w języku Java, nawigację wewnątrz drzew DOM w języku Java, wprowadzenie do języka zapytań XPath ułatwiającego przeszukiwanie drzew DOM oraz parsery DOM, służące do automatycznego generowania drzew DOM na podstawie dokumentów XML.

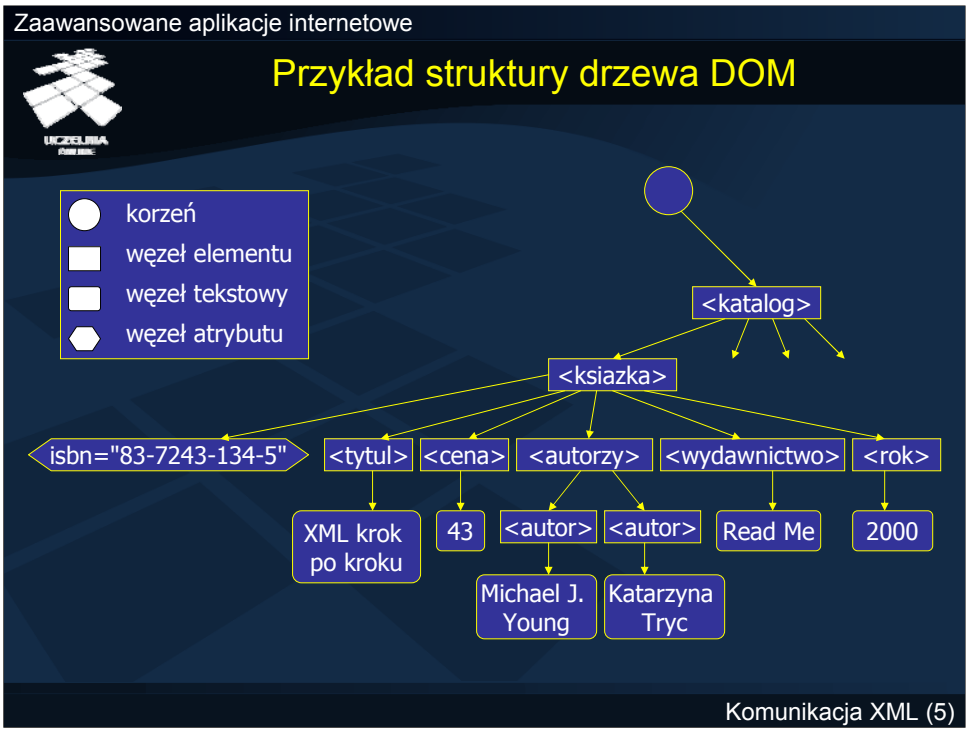


Język XML umożliwia realizację elektronicznej wymiany danych pomiędzy niezależnymi, heterogenicznymi systemami informatycznymi. O ile jednak generowanie dokumentów XML jest nieskomplikowanym zadaniem programistycznym, o tyle składniowy rozbiór takich dokumentów stanowi istotne wyzwanie dla twórców aplikacji. W celu ułatwienia implementacji aplikacji przetwarzających dokumenty XML zaproponowano, aby programiści posługiwali się alternatywnym modelem danych i udostępniono im specjalizowane biblioteki programistyczne umożliwiające dwukierunkową konwersję z- i do- formatu XML.

Document Object Model (DOM) jest uniwersalnym standardem modelowania dokumentów XML przy użyciu struktury drzewa – znaczniki XML i ich zawartość są modelowane przez węzły drzewa, a zagnieżdżanie znaczników służy za podstawę do konstruowania hierarchii. W praktyce Document Object Model jest wykorzystywany jako forma reprezentacji dokumentów XML w pamięci komputera. Transformacja dokumentu XML do postaci Document Object Model jest realizowana automatycznie przez specjalny moduł, nazywany parserem DOM. Implementacja, adresowanie i przeszukiwanie drzew Document Object Model mogą być realizowane przy użyciu biblioteki o nazwie DOM API. Ponadto, zaawansowane operacje przeszukiwania drzew DOM mogą być wyrażane w specjalizowanym języku zapytań XPath, implementowanym przez biblioteki DOM API.

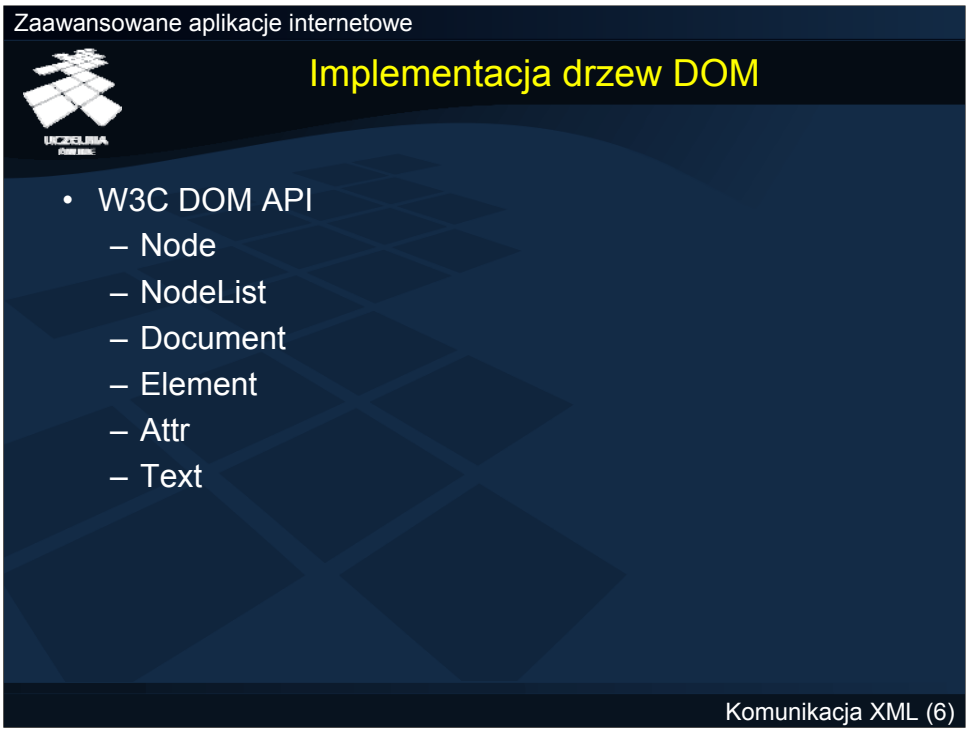


W celu przedstawienia własności modelu DOM podczas całego wykładu będziemy posługiwać się tym samym przykładowym dokumentem XML przedstawionym na slajdzie. Dokument ten reprezentuje fragment katalogu książek sprzedawanych przez księgarnię internetową. Każda książka opisana jest za pomocą znaczników <ksiazka>, zawierających tytuł książki, otoczony znacznikami <tytul>, jej cenę, otoczoną znacznikami <cena>, listę autorów, otoczoną znacznikami <autorzy> i <autor>, nazwę wydawnictwa, otoczoną znacznikami <wydawnictwo> i rok wydania, otoczony znacznikami <rok>. Numer ISBN książki jest zapisany jako atrybut "isbn" znacznika <książka>. Cały katalog jest otoczony znacznikami <katalog>.



Strukturę drzewa DOM dla przedstawionego wcześniej dokumentu XML zamieszczono na slajdzie. Drzewo DOM składa się m.in. z węzłów czterech rodzajów: korzenia - stanowiącego początek drzewa, węzłów elementu - reprezentujących pary znaczników XML, węzłów tekstowych - reprezentujących tekst pomiędzy znacznikiem otwierającym a zamykającym, węzłów atrybutu - reprezentujących atrybuty znaczników. Węzły te są ze sobą połączone krawędziami wynikającymi z relacji zagnieżdżania znaczników w oryginalnym dokumencie.

Takie odwzorowanie dokumentu XML w pamięci komputera istotnie upraszcza przetwarzanie jego treści. Programista może swobodnie nawigować w strukturze drzewa i w dowolnej kolejności odczytywać elementarne dane źródłowe.



W celu ułatwienia implementacji drzew DOM, organizacja W3C zaproponowała specyfikację biblioteki programisty zawierającej predefiniowane typy danych oraz funkcje nawigacyjne i dostępowe. Specyfikacja ta nazywa się W3C DOM API i jest obecnie podstawą, na której opierają się konkretne biblioteki programisty XML oferowane dla rozmaitych języków programowania (np. org.w3c.dom dla języka Java). Oto najważniejsze typy danych reprezentujące składniki drzew DOM:

• Node reprezentuje węzeł w drzewie DOM (węzeł elementu, węzeł tekstowy, itd.)

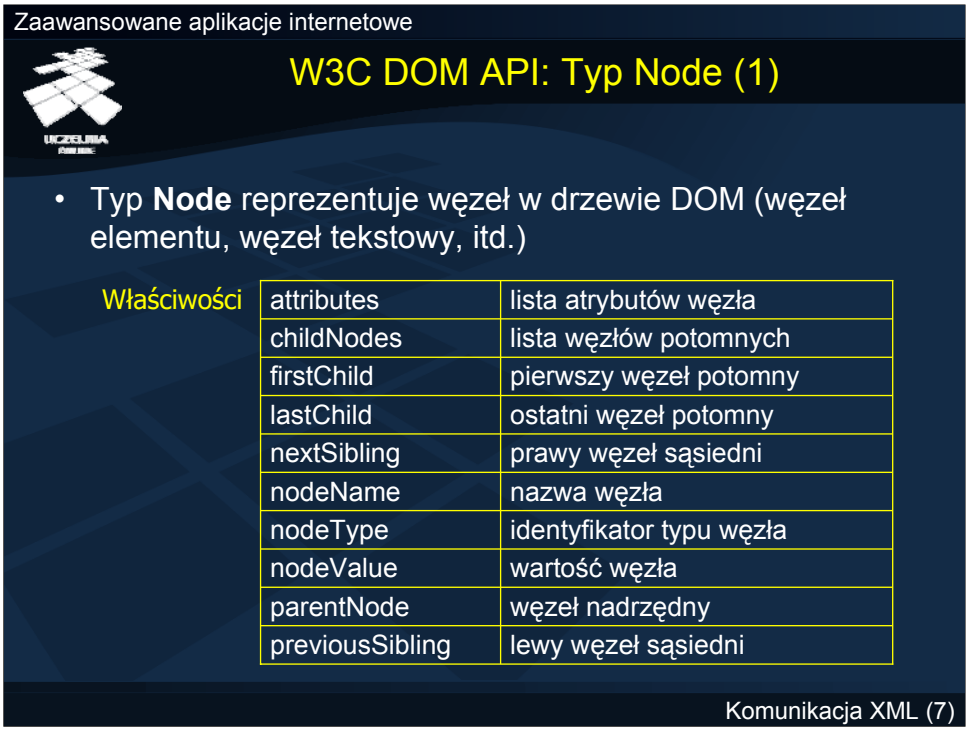
• NodeList reprezentuje listę obiektów typu Node

• Document modeluje całe drzewo DOM; wszystkie węzły drzewa są jego potomkami

• Element modeluje węzeł reprezentujący parę znaczników XML

• Attr reprezentuje atrybut znacznika XML w formie tzw. węzła atrybutu

• Text reprezentuje treść umieszczoną wewnątrz pary znaczników XML



Na slajdzie przedstawiono nazwy wybranych właściwości opisujących każdy obiekt typu Node. Najważniejsze z nich to:

•nodeValue, opisująca wartość węzła, np. tytuł książki,

•nodeName, opisująca nazwę węzła, np. "cena",

•childNodes, będąca listą węzłów potomnych w stosunku do bieżącego węzła,

•attributes, będąca listą węzłów atrybutowych związanych z bieżącym węzłem.

W3C DOM API: Typ Node (2)

- Typ **Node** reprezentuje węzeł w drzewie DOM (węzeł elementu, węzeł tekstowy, itd.)

Operacje

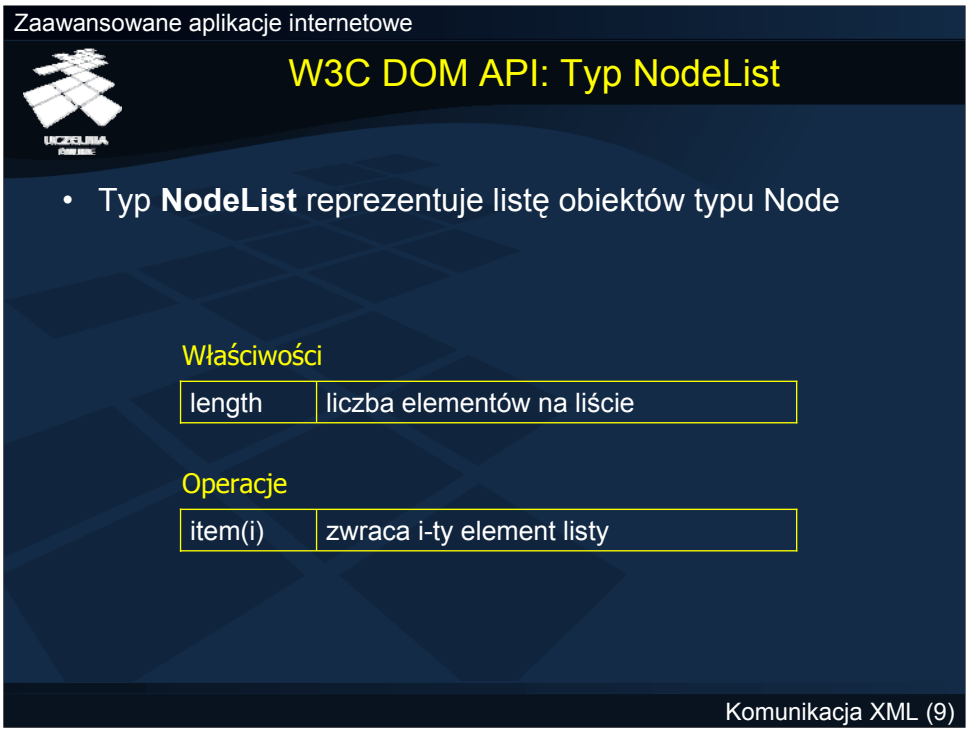
appendChild(n)	dołącza nowy węzeł jako ostatni węzeł potomny
cloneNode(b)	zwraca kopię węzła z/bez węzłami potomnymi
hasChildNodes()	zwraca prawdę, jeżeli węzeł zawiera węzły potomne
insertBefore(n,n)	dołącza nowy węzeł jako węzeł potomny przed wskazanym węzłem
removeChild(n)	usuwa wskazany węzeł potomny
replaceChild(n,n)	zamienia istniejący węzeł potomny z podanym węzłem

Na slajdzie przedstawiono nazwy wybranych operacji oferowanych przez każdy obiekt typu Node. Najważniejsze z nich to:

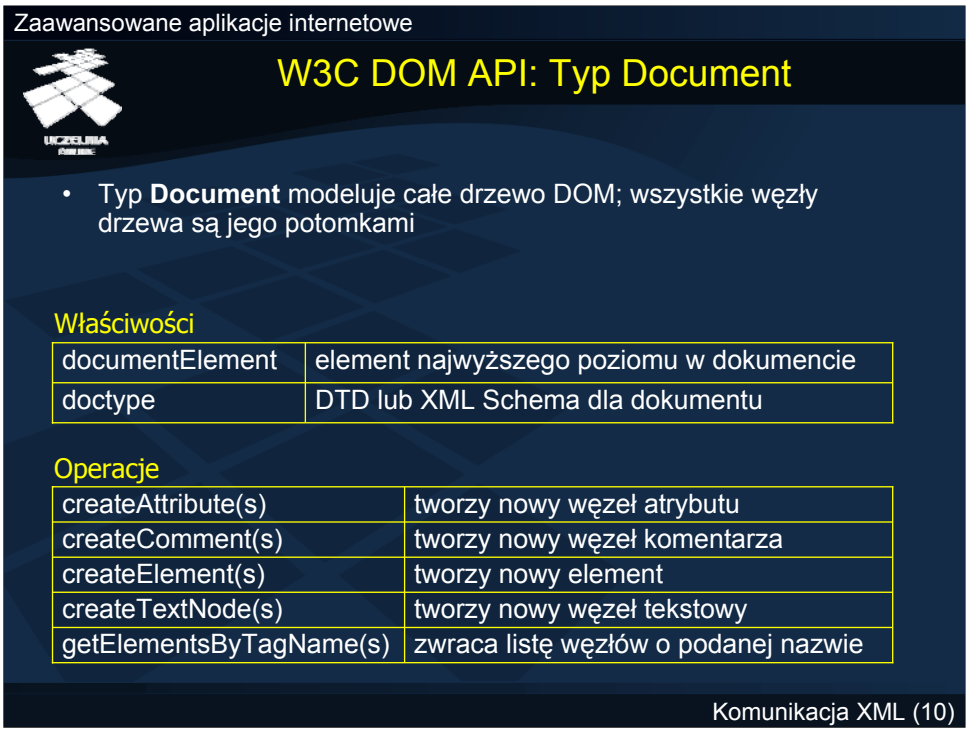
•appendChild(), umożliwiająca dołączenie nowego węzła potomnego do bieżącego węzła,

•removeChild(), umożliwiająca odłączenie wybranego węzła potomnego od bieżącego węzła,

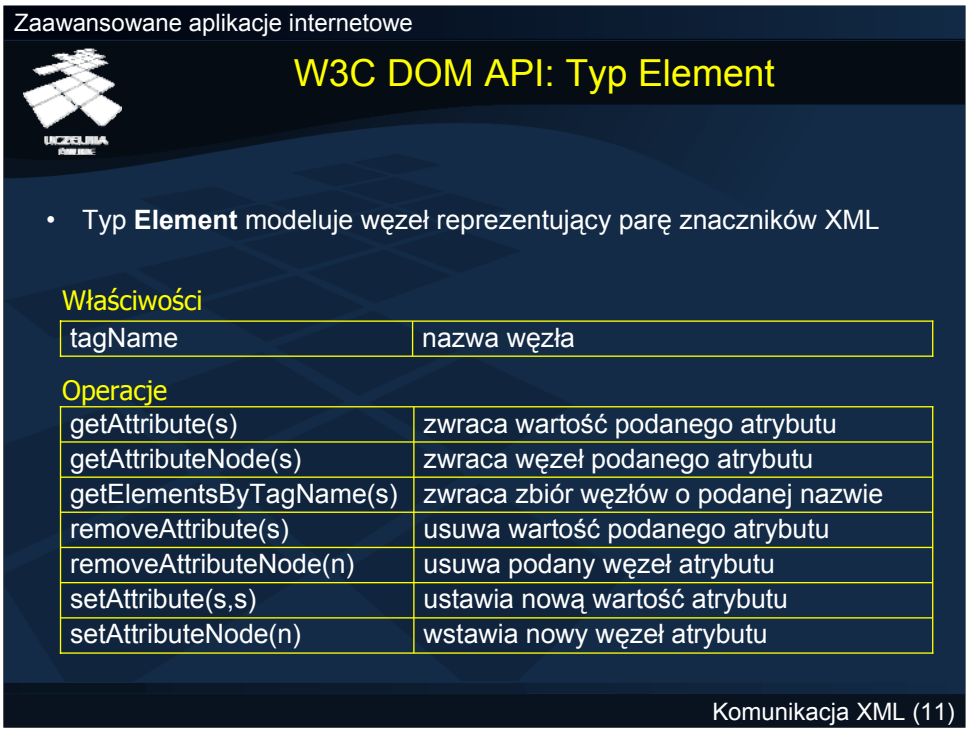
•hasChildNodes(), umożliwiająca sprawdzenie, czy bieżący węzeł posiada węzły potomne.



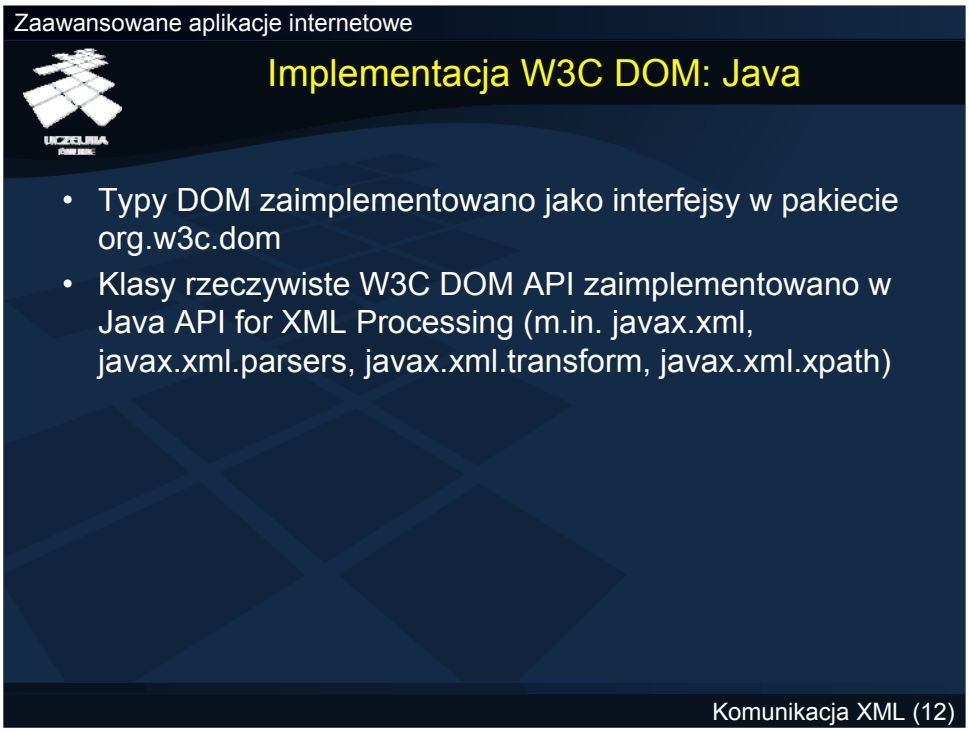
Na slajdzie przedstawiono wybrane właściwości i operacje oferowane przez obiekty typu NodeList. Właściwość "length" umożliwia odczyt liczby elementów na liście, a operacja "item(i)" daje dostęp do i-tego elementu listy.



Na slajdzie przedstawiono wybrane właściwości i operacje oferowane przez obiekty typu Document. Właściwość "documentElement" reprezentuje węzeł elementu najwyższego poziomu (np. <katalog>). Operacja "createElement()" umożliwia utworzenie nowego węzła w drzewie DOM. Operacja "getElementsByTagName()" przeszukuje drzewo DOM i zwraca listę węzłów o podanej nazwie.

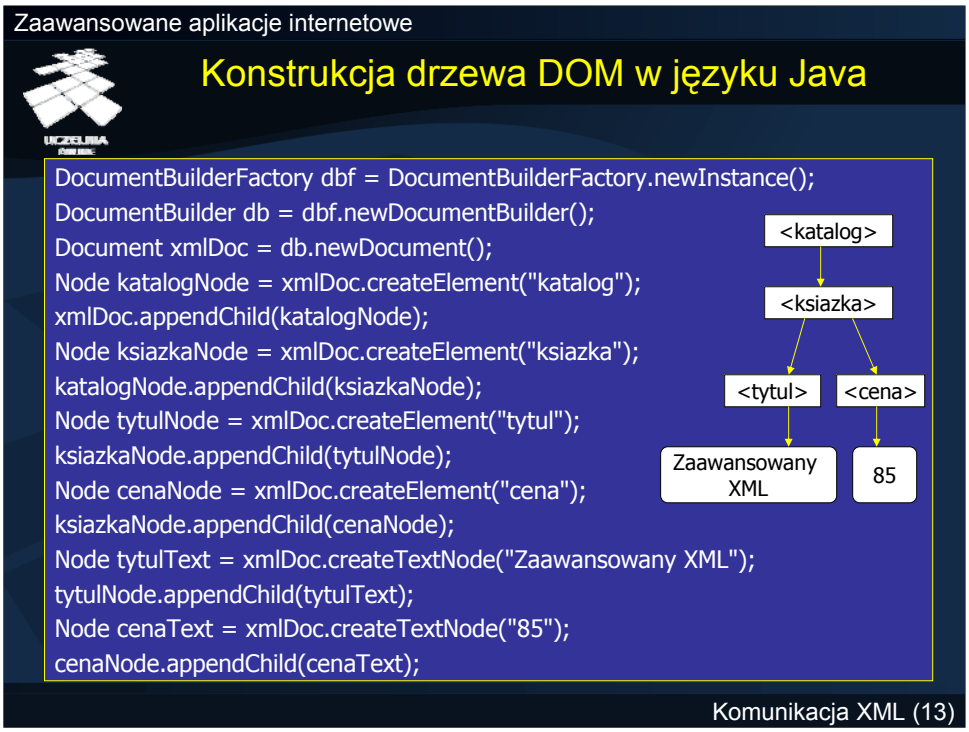


Na slajdzie przedstawiono wybrane właściwości i operacje oferowane przez obiekty typu Element. Właściwość "tagName" opisuje nazwę węzła (znacznika), operacja "getAttributeNodes()" zwraca listę atrybutów powiązanych ze znacznikiem XML.



Specyfikacja W3C DOM API wyznacza ogólne wymogi dla implementacji bibliotek programistycznych służących do przetwarzania dokumentów XML. Biblioteki takie powstały i powstają dla wielu różnorodnych języków programowania. W języku Java dostępna jest biblioteka Java API for XML Processing (JAXP), która jest kompatybilna z W3C DOM API. Biblioteka JAXP zawiera m.in. następujące pakiety:

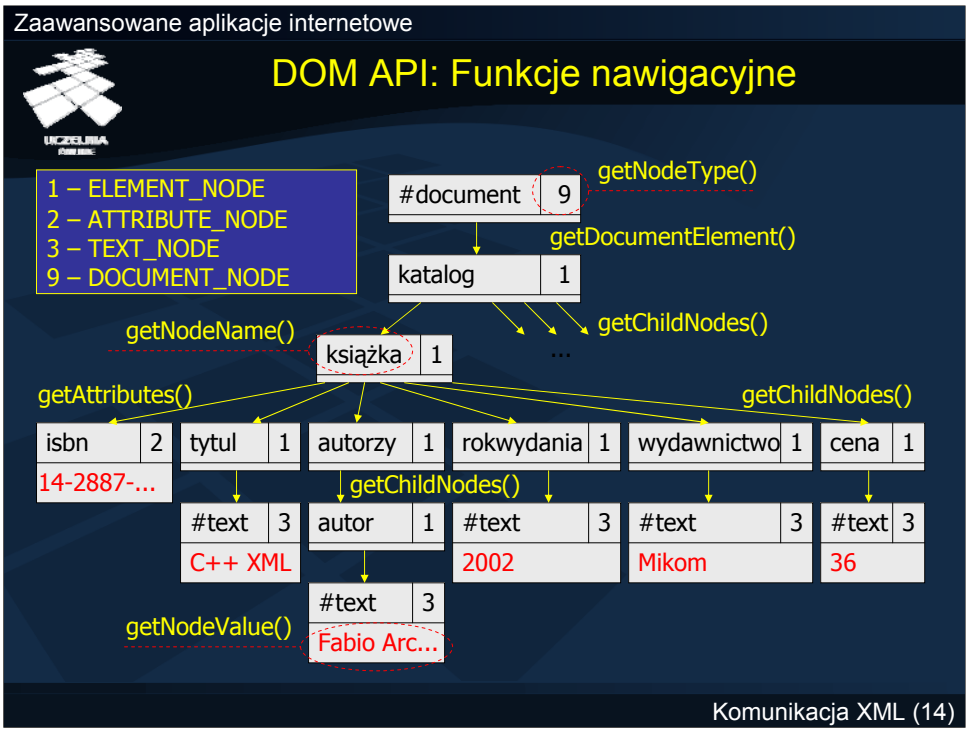
•javax.xml
•javax.xml.datatype
•javax.xml.namespace
•javax.xml.parsers
•javax.xml.transform
•javax.xml.transform.dom
•javax.xml.transform.sax
•javax.xml.transform.stream
•javax.xml.validation
•javax.xml.xpath
•org.w3c.dom
•org.w3c.dom.bootstrap
•org.w3c.dom.events
•org.w3c.dom.ls
•org.w3c.dom.ranges
•org.w3c.dom.traversal
•org.xml.sax
•org.xml.sax.ext
•org.xml.sax.helpers



Na slajdzie przedstawiono kod programu Java, który buduje w pamięci operacyjnej strukturę drzewa DOM zgodną z przedstawionym rysunkiem.

Do utworzenia nowego drzewa DOM służy metoda newDocument() obiektu klasy DocumentBuilder, który z kolei jest tworzony za pomocą metody newDocumentBuilder() obiektu klasy DocumentBuilderFactory, który wreszcie jest tworzony za pomocą metody newInstance(). Drzewo DOM jest reprezentowane przez obiekt interfejsu Document, nazwany w przykładzie "xmlDoc".

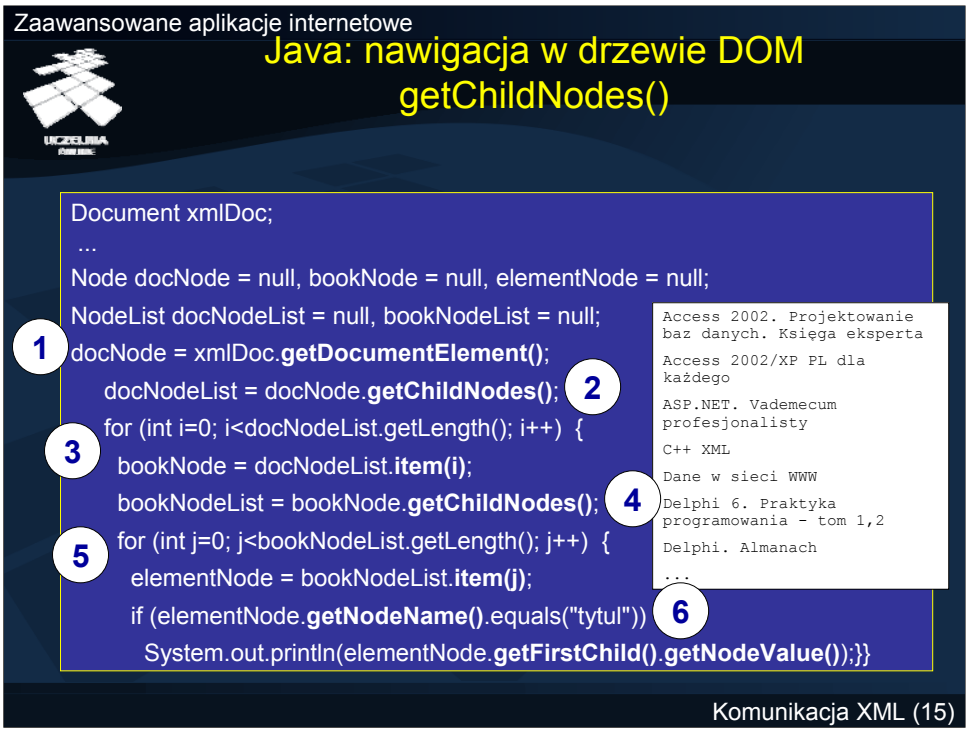
W dalszej części programu tworzony jest węzeł elementu "katalog", reprezentowany przez obiekt interfejsu Node. Węzeł ten jest dołączany do korzenia drzewa DOM za pomocą metody appendChild(). Następnie tworzymy węzły elementów "książka", "tytuł" oraz "cena" i odpowiednio łączymy je w strukturze drzewa. W ostatniej części programu tworzone są dwa węzły tekstowe i są one dołączane do wcześniej utworzonych węzłów elementów "tytuł" i "cena".



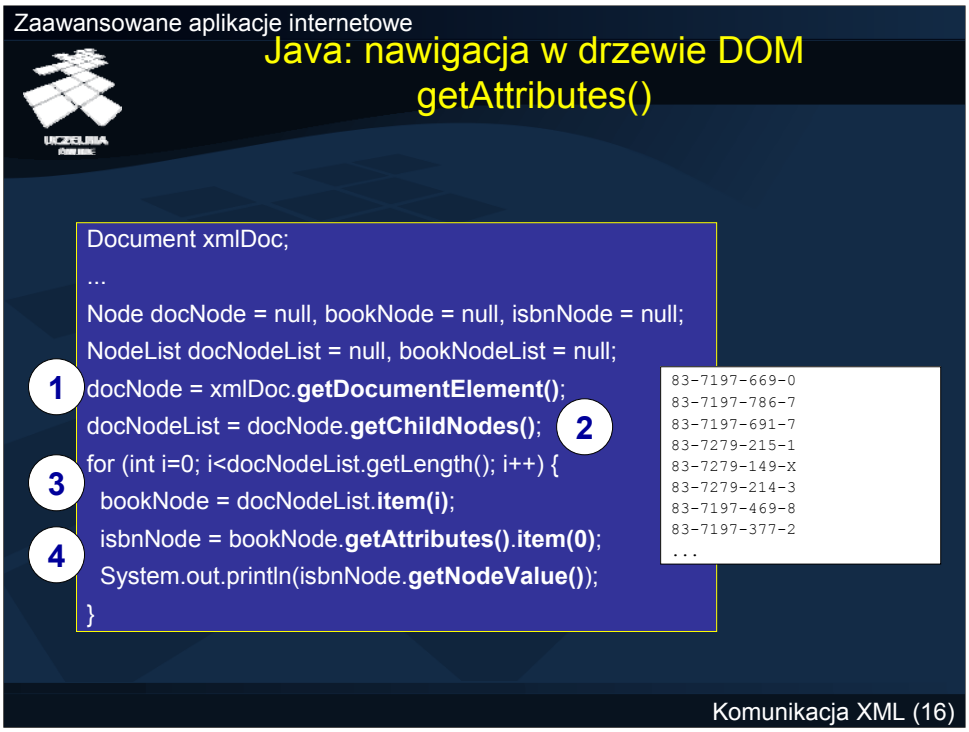
Nawigacja wewnątrz drzew DOM odbywa się przy użyciu metod zgodnych ze specyfikacją W3C DOM API. Na slajdzie przedstawiono szczegółowo przykładową strukturę drzewa DOM wraz z nazwami wybranych metod nawigacyjnych.

Każdy węzeł drzewa posiada trzy właściwości: nazwę, wartość i typ. Na przedstawionym rysunku nazwa węzła znajduje się w lewym górnym rogu węzła, typ - w prawym górnym rogu, a wartość - w dolnej części węzła. Do odczytu nazwy węzła służy metoda getNodeName(), do odczytu typu węzła - metoda getNodeType(), a do odczytu wartości węzła wykorzystujemy metodę getNodeValue(). Metoda getNodeType() zwraca numeryczny identyfikator typu węzła. Na slajdzie przedstawiono wykaz wybranych kodów typów węzłów, np. wartość 3 oznacza węzeł tekstowy. Zauważmy również, że nie każdy węzeł posiada wartość, a nazwy węzłów tekstowych i korzenia to zawsze "#text" i "#document".

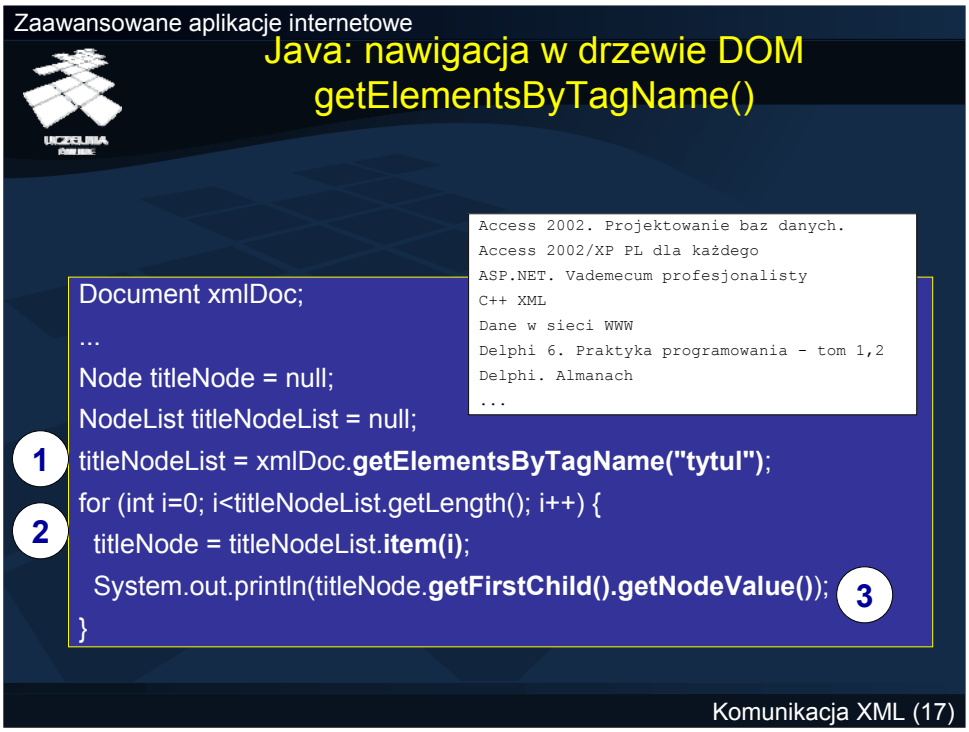
Do przechodzenia od węzła nadrzędnego do węzłów podrzędnych służy metoda getChildNodes(), lecz z pewnymi wyjątkami. Otóż przejście z korzenia drzewa do węzła elementu najwyższego poziomu odbywa się z użyciem metody getDocumentElement(), a do przechodzenia do węzłów atrybutowych służy metoda getAttributes().



Na slajdzie przedstawiono przykład programu Java, który w oparciu o przedstawiony wcześniej dokument XML wyświetla tytuły wszystkich książek z katalogu. Oto najważniejsze fragmenty kodu źródłowego:
1. W zmiennej docNode zapisujemy węzeł elementu najwyższego poziomu (znacznik <katalog>).
2. W zmiennej docNodeList zapisujemy listę węzłów podrzędnych w stosunku do węzła docNode (znaczniki <ksiazka>).
3. Iterując po węzłach z listy docNodeList pobieramy kolejne jej elementy do zmiennej bookNode.
4. W zmiennej bookNodeList zapisujemy listę węzłów podrzędnych w stosunku do węzła bookNode (znaczniki <tytul>, <autorzy>, <rok_wydania>, <wydawnictwo>, <cena>).
5. Iterując po węzłach z listy bookNodeList pobieramy kolejne jej elementy do zmiennej elementNode.
6. Sprawdzamy, czy węzeł elementu reprezentuje znacznik <tytul>. Jeśli tak, to wyświetlamy na ekranie wartość pierwszego węzła podrzędnego w stosunku do węzła elementNode, który powinien być węzłem tekstowym zawierającym tytuł książki.



Na slajdzie przedstawiono przykład programu Java, który w oparciu o przedstawiony wcześniej dokument XML wyświetla wartości pierwszego atrybutu każdego znacznika <ksiazka> (czyli numery ISBN książek z katalogu). Oto najważniejsze fragmenty kodu źródłowego:
1. W zmiennej docNode zapisujemy węzeł elementu najwyższego poziomu (znacznik <katalog>).
2. W zmiennej docNodeList zapisujemy listę węzłów podrzędnych w stosunku do węzła docNode (znaczniki <ksiazka>).
3. Iterując po węzłach z listy docNodeList pobieramy kolejne jej elementy do zmiennej bookNode.
4. W zmiennej isbnNode zapisujemy pierwszy element z listy węzłów atrybutowych podrzędnych w stosunku do węzła bookNode. Węzeł isbnNode reprezentuje atrybut "isbn" znacznika <ksiazka>. Jego wartość wyświetlamy na ekranie.

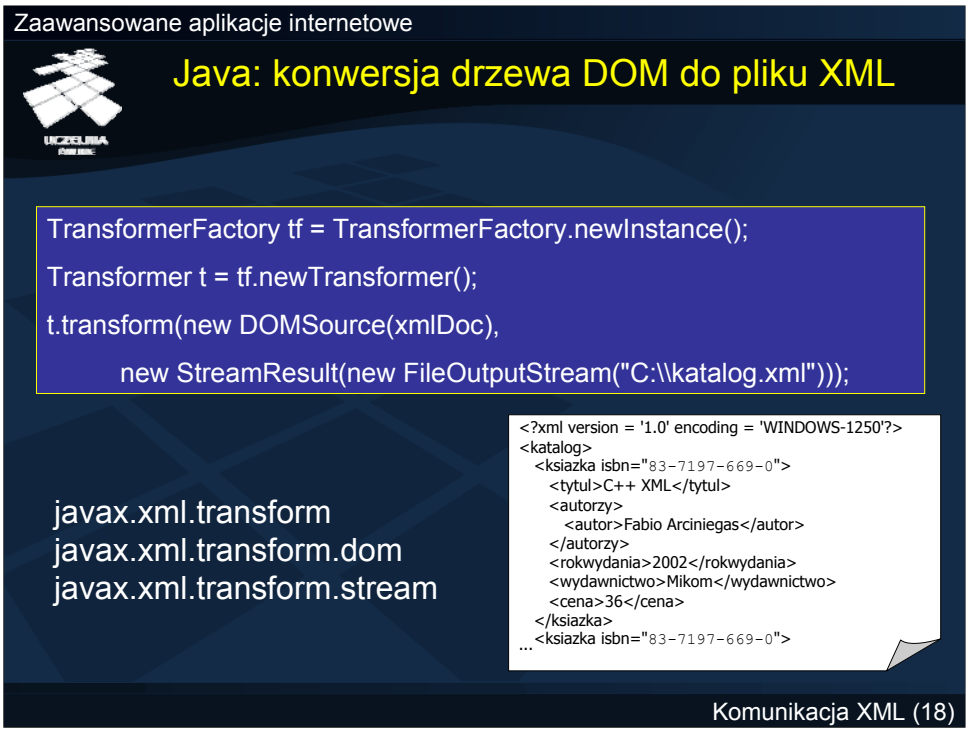


Na slajdzie przedstawiono przykład programu Java, który w oparciu o przedstawiony wcześniej dokument XML wyświetla tytuły wszystkich książek opisanych w katalogu. W przeciwieństwie do wcześniejszego przykładu, ten program wykorzystuje przydatną metodę getElementsByTagName(). Oto najważniejsze fragmenty kodu źródłowego:

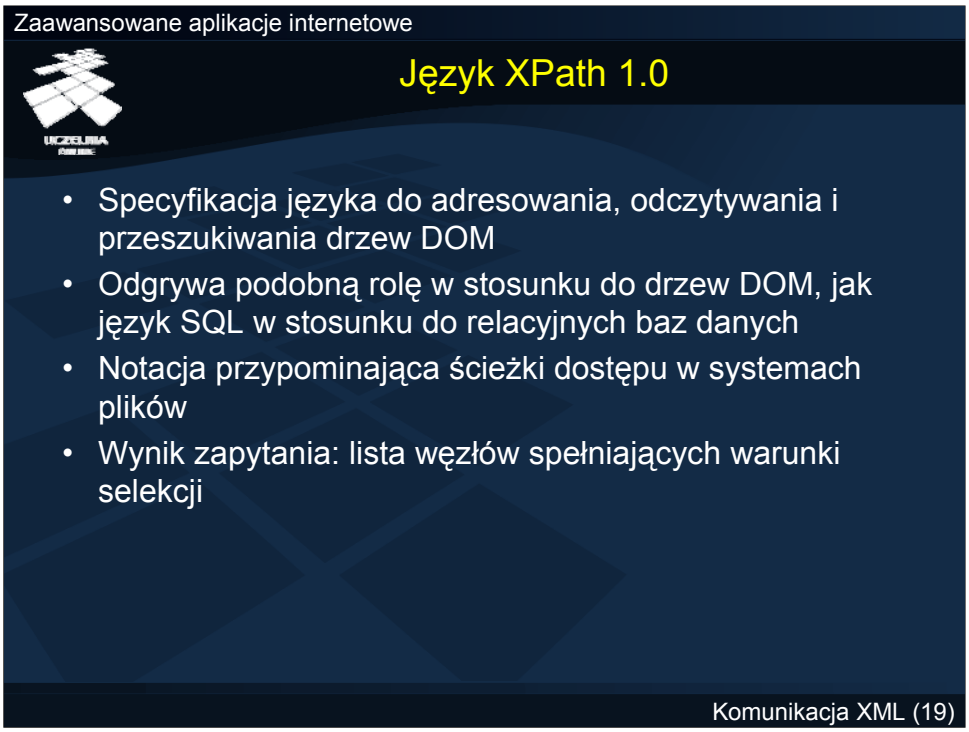
1. W zmiennej titleNodeList zapisujemy listę wszystkich węzłów o nazwie "tytul". Przeszukanie całego drzewa DOM jest realizowane przez metodę getElementsByTagName().

2. Iterując po węzłach z listy titleNodeList pobieramy kolejne jej elementy do zmiennej titleNode.

3. Wyświetlamy na ekranie wartość pierwszego węzła podrzędnego w stosunku do węzła titleNode, który powinien być węzłem tekstowym zawierającym tytuł książki.



Drzewo DOM utworzone w pamięci operacyjnej może zostać przekształcone do postaci odpowiadającego mu dokumentu XML. Biblioteka JAXP zawiera klasy TransformerFactory i Transformer, które generują wyjściowy strumień znakowy XML na podstawie źródłowego drzewa DOM. Na slajdzie przedstawiono przykładowy fragment programu Java, który na podstawie drzewa DOM reprezentowanego przez zmienną "xmlDoc" zapisuje na dysku plik XML o nazwie katalog.xml. Klasy biblioteczne wykorzystane w przykładzie pochodzą z pakietów javax.xml.transform, javax.xml.transform.dom i javax.xml.transform.stream.



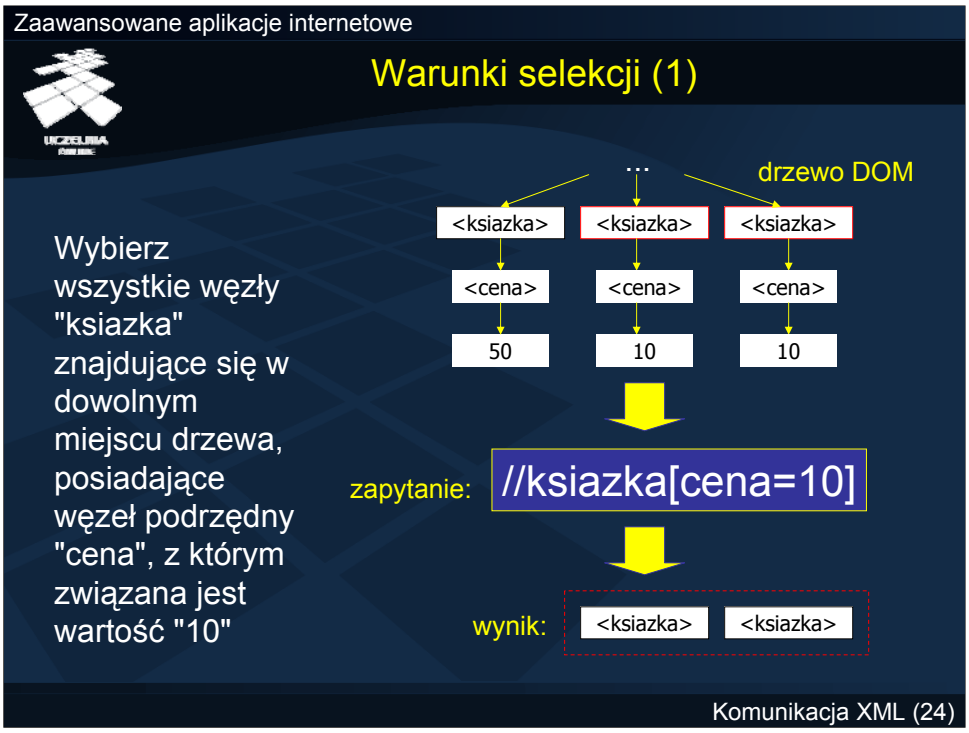
Przeszukiwanie złożonych drzew DOM wyłącznie za pomocą metod nawigacyjnych prowadzi do istotnego skomplikowania kodu aplikacji, przejawiającego się dużą liczbą zagnieżdżonych pętli oraz warunków logicznych. Interesującym rozwiązaniem pozwalającym na uproszczenie tego typu problemów jest technologia XPath. XPath to specyfikacja języka wyrażeń służącego do adresowania, odczytywania i przeszukiwania drzew DOM reprezentujących dokumenty XML. XPath odgrywa podobną rolę w stosunku do drzew DOM, jak język SQL w stosunku do relacyjnych baz danych, tzn. pozwala formułować deklaratywne zapytania, które są wykonywane przez oprogramowanie systemowe. XPath stosuje notację przypominającą ścieżki dostępu w systemach plików - zapytanie XPath jest łańcuchem znakowym zawierającym wiele ukośników. Wynikiem ewaluacji zapytania XPath jest lista węzłów spełniających warunki selekcji.

Podstawową formą zapytań XPath są wyrażenia ścieżkowe. Wyrażenie ścieżkowe wybiera węzły drzewa poprzez opisanie absolutnej lub względnej ścieżki do nich prowadzącej. Przykład wyrażenia ścieżkowego przedstawiono na slajdzie. Wyrażenie to jest łudząco podobne do ścieżki dostępu do pliku w systemie plików. Najistotniejszą różnicą jest jednak to, że o ile w systemie plików nazwy obiektów znajdujących się w jednym katalogu nie mogą się powtarzać, o tyle w drzewie DOM węzeł może posiadać dowolnie wiele węzłów podrzędnych nazwanych jednakowo. Stąd wyrażenie ścieżkowe XPath może wybierać wiele węzłów, a nie tylko jeden. Wynikiem przedstawionego na slajdzie zapytania będzie lista węzłów "ksiazka" podrzędnych w stosunku do węzłów/węzła "katalog", które z kolei są podrzędnymi w stosunku do korzenia drzewa DOM.

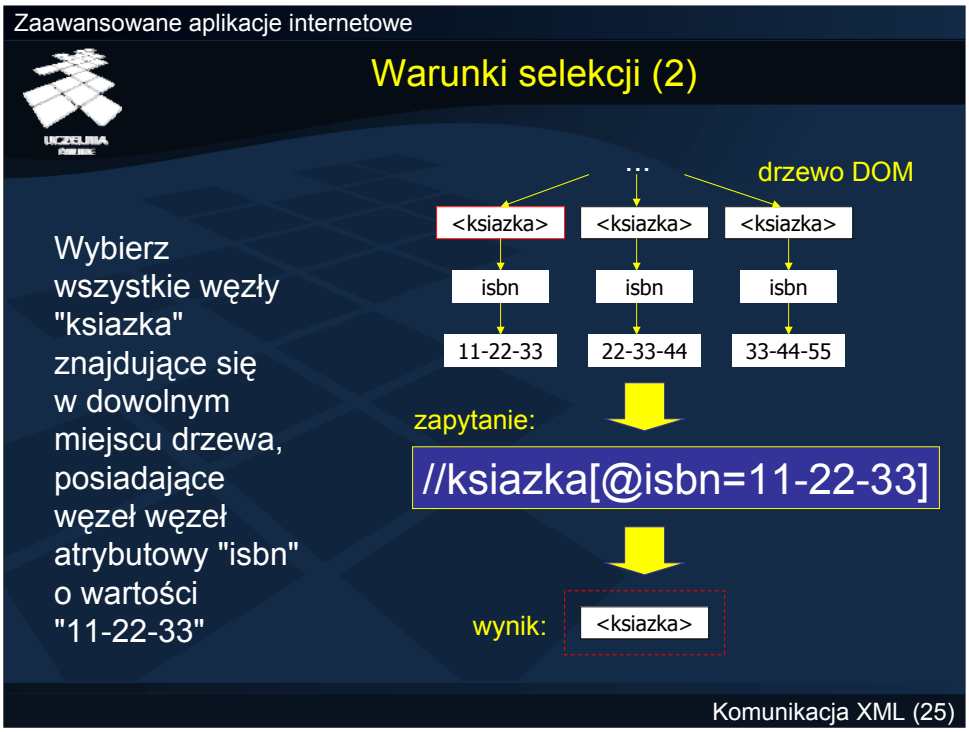
Podwójny ukośnik służy do skrótowego zapisu dowolnie długiej ścieżki w drzewie. Zapytanie XPath przedstawione na slajdzie zwróci listę wszystkich węzłów "tytul" znajdujących się w dowolnym miejscu w drzewie DOM.

Znak "*" służy do wybierania węzłów o dowolnej nazwie. Przedstawione na slajdzie przykładowe zapytanie XPath zwróci wszystkie węzły podrzędne w stosunku do węzłów "ksiazka" znajdujących się w dowolnym miejscu drzewa DOM.

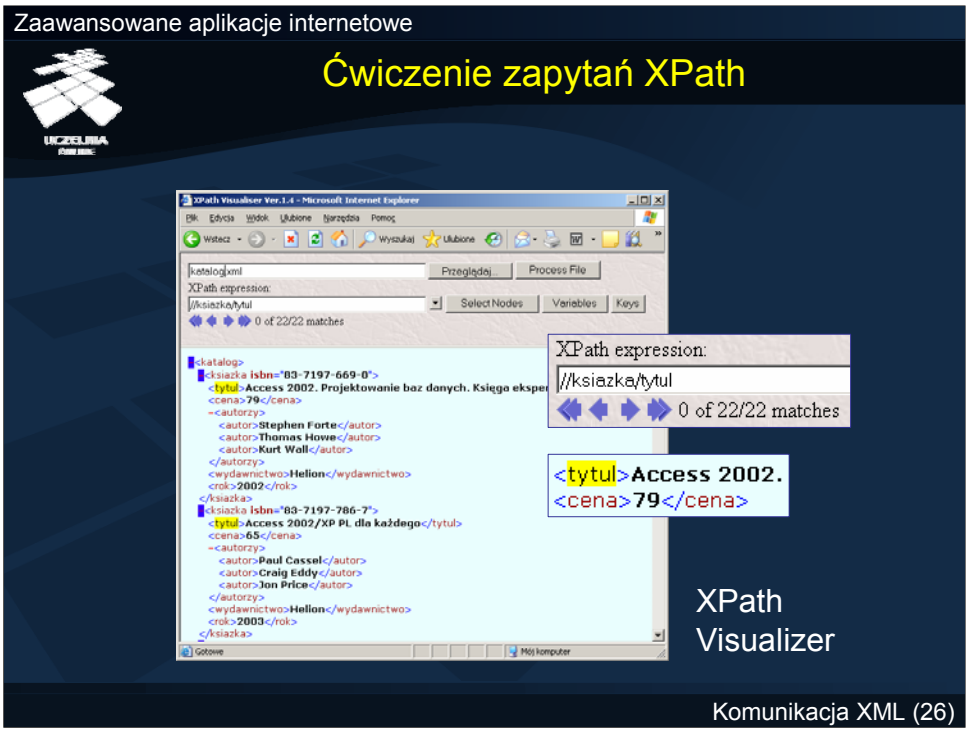
Znaki "[]" umożliwiają wyodrębnienie pojedynczego węzła spośród węzłów podrzędnych. Podana w nawiasach liczba jest numerem kolejnym węzła w jego poddrzewie. Zapytanie XPath przedstawione na slajdzie wybierze pierwsze węzły "autor" z każdego poddrzewa utworzonego przez węzły "autorzy", znajdujące się w dowolnym miejscu drzewa DOM.



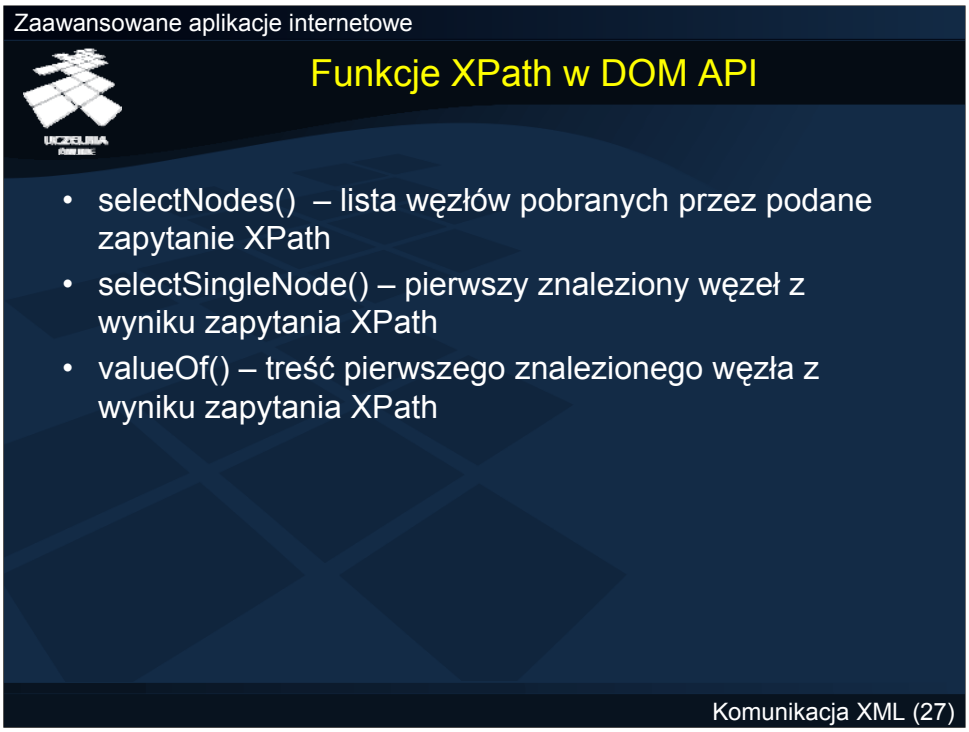
Język XPath umożliwia selekcję węzłów w oparciu o warunki logiczne. Warunki selekcji zapisuje się w prostokątnych nawiasach przy nazwie węzłów, które podlegają selekcji. Przykładowe zapytanie XPath przedstawione na slajdzie wybiera wszystkie te węzły "ksiazka", które posiadają węzeł podrzędny "cena", który z kolei posiada podrzędny węzeł tekstowy o wartości "10".



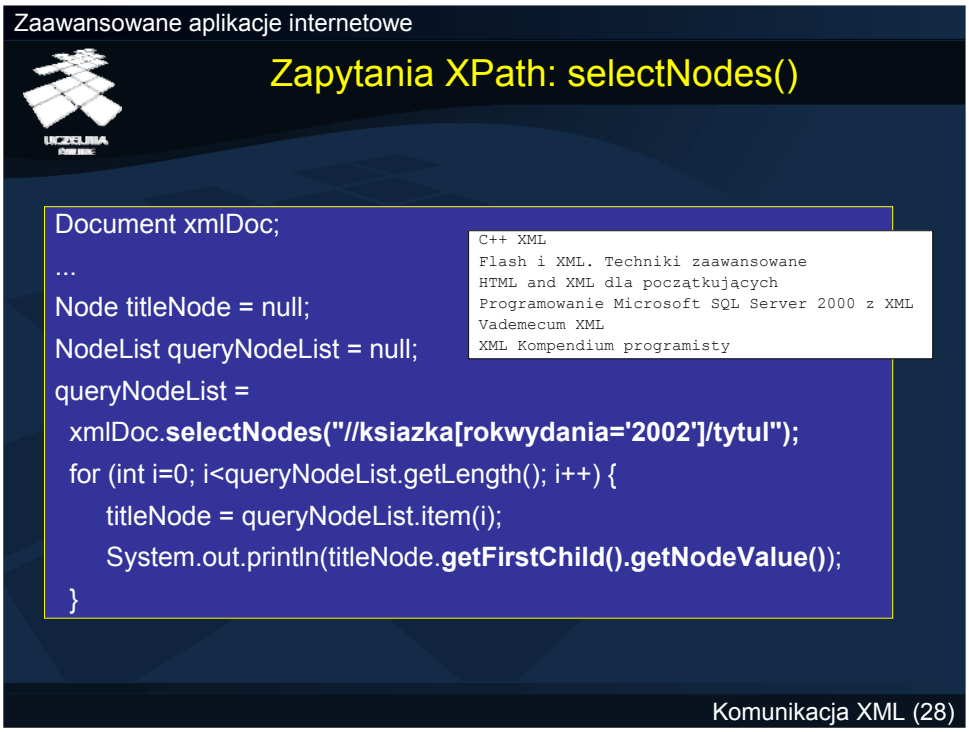
Na slajdzie przedstawiono przykład zapytania XPath dokonującego selekcji w oparciu o wartość węzła atrybutowego. Nazwę węzła atrybutowego poprzedzamy znakiem "@". Wynikiem zapytania będą wszystkie węzły "ksiazka" posiadające atrybut "isbn" o wartości "11-22-33".



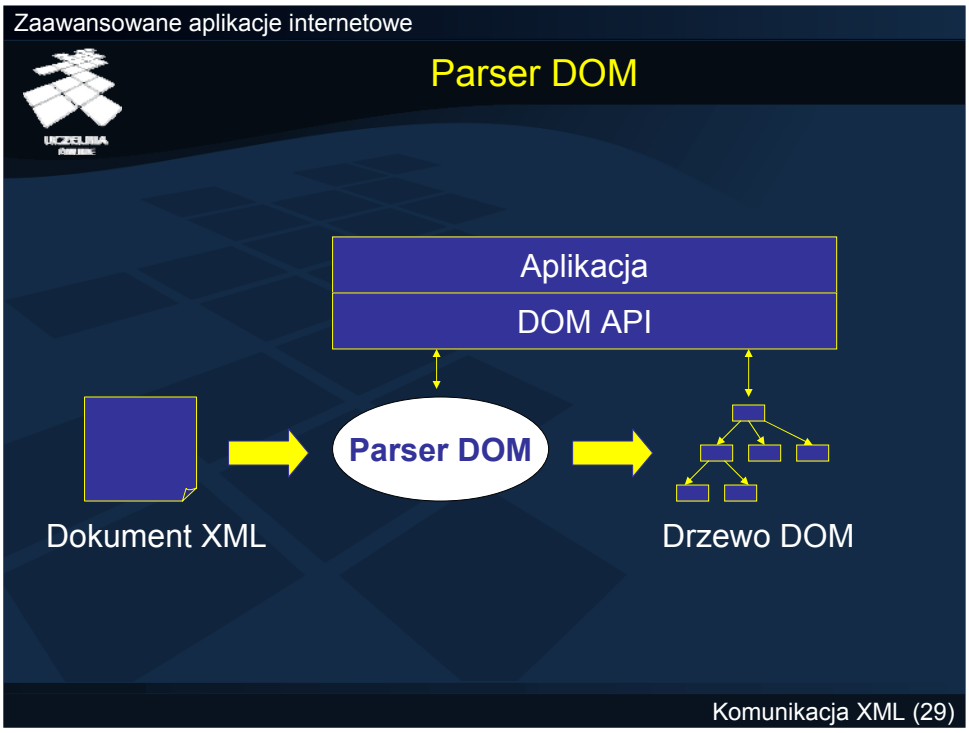
Swoje umiejętności definiowania zapytań w języku XPath łatwo jest rozwijać za pomocą prostego narzędzia XPath Visualizer (http://www.topxml.com/xpathvisualizer/default.asp), umożliwiającego załadowanie wybranego dokumentu XML i wykonywanie na nim zapytań ad-hoc. Narzędzie XPath Visualizer pracuje na platformie przeglądarki Microsoft Internet Explorer.



Na silną pozycję języka XPath wpływa to, że specyfikacja W3C DOM API przewiduje operacje realizacji zapytań XPath na drzewach DOM. Programista Java może w nieskomplikowany sposób korzystać z metod selectNodes(), selectSingleNode() i valueOf(), które pobierają treść zapytania i generują jego wynik. Metoda selectNodes() zwraca listę węzłów będących wynikiem zapytania XPath. Metoda selectSingleNode() zwraca pierwszy (najczęściej jedyny) węzeł będący wynikiem zapytania XPath. Metoda valueOf() zwraca treść pierwszego potomka pierwszego węzła będącego wynikiem zapytania XPath. Na kolejnym slajdzie przedstawimy przykład wykorzystania zapytań XPath.

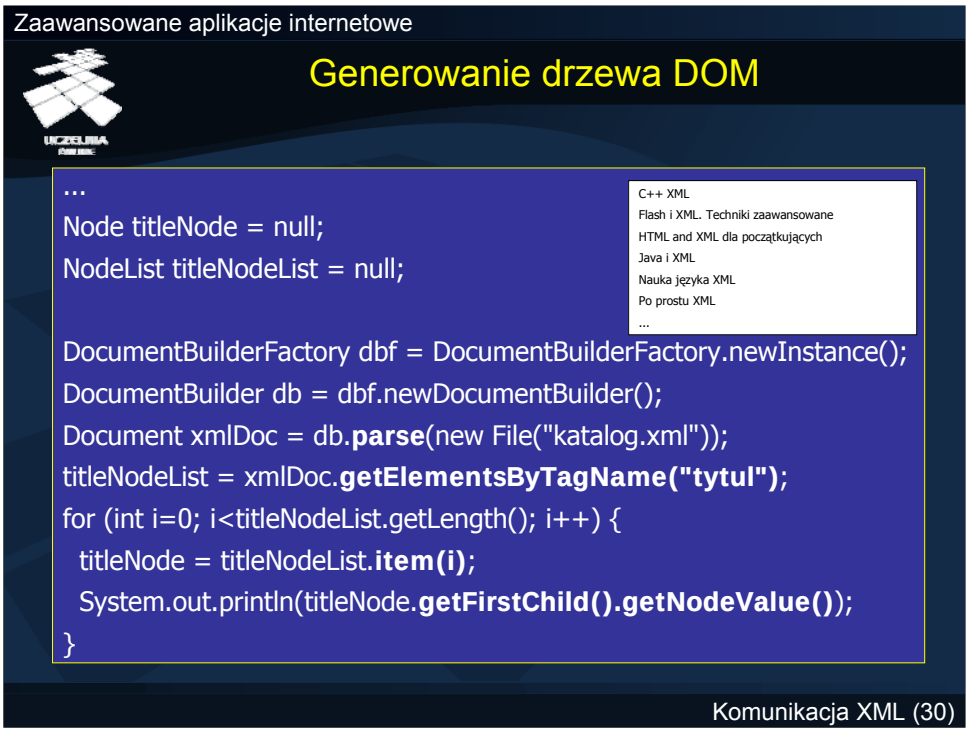


Przykładowy program Java przedstawiony na slajdzie przeszukuje drzewo DOM w celu znalezienia wszystkich węzłów "tytul" opisujących książki wydane w roku 2002. Stosowne zapytanie XPath stanowi argument wywołania metody selectNodes() obiektu xmlDoc, reprezentującego drzewo DOM dokumentu XML. Wynikiem metody selectNodes() jest lista węzłów spełniających warunki zapytania. W dalszej części programu iterujemy po wszystkich węzłach z wygenerowanej listy i wyświetlamy na ekranie wartości ich pierwszych potomków, czyli węzłów tekstowych zawierających brzmienie tytułu książki.



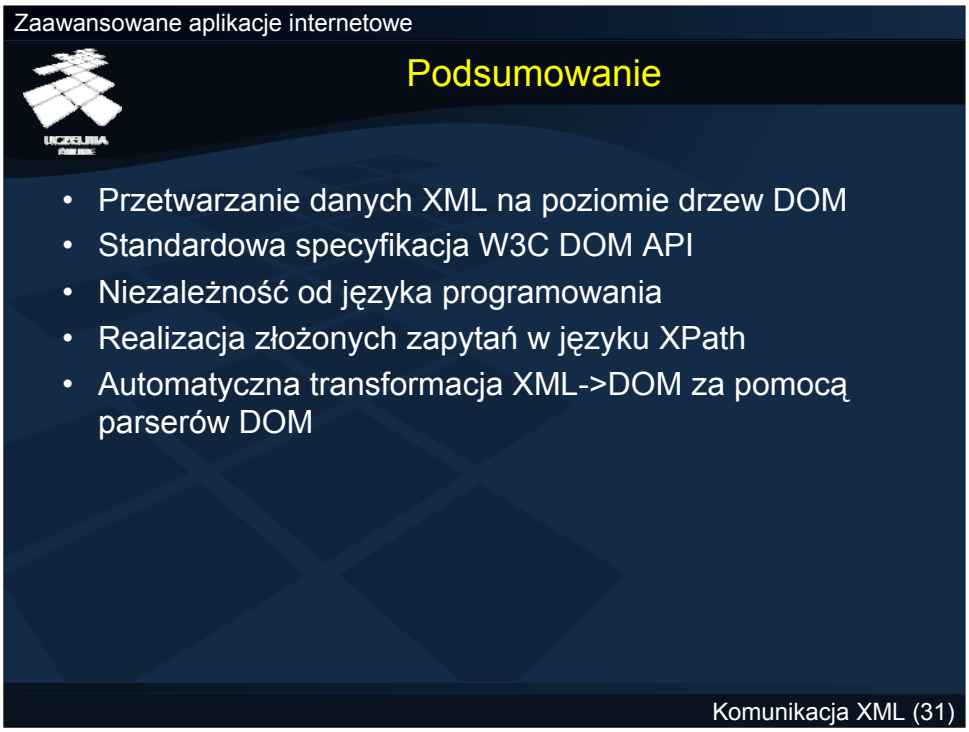
Istotą wykorzystywania modelu DOM do reprezentacji i przetwarzania dokumentów XML jest to, aby drzewa DOM mogły być budowane automatycznie w oparciu o treść plików XML. Programista powinien być jak najmniej zaangażowany w kwestie transformacji danych pomiędzy formatami XML i DOM. Jeżeli ten warunek zostanie spełniony, to będziemy mogli przyjąć, że plik XML jest po prostu formatem zapisu drzewa DOM na dysku, a logika przetwarzania danych w aplikacji będzie mogła być skupiona wokół drzewiastej reprezentacji informacji.

Do automatycznej transformacji plików XML do struktur drzew DOM służą parsery DOM. Parsery DOM wchodzą w skład bibliotek programistycznych (np. JAXP) i mogą być wykorzystywane w różnorodnych językach programowania. Na slajdzie przedstawiono ogólny diagram przepływu danych w aplikacji przetwarzającej dane XML. Plik dokumentu XML trafia na wejście modułu parsera DOM, gdzie jest transformowany do struktury drzewa DOM, będącego wynikiem pracy parsera DOM. Sterowanie parserem DOM oraz dalsze przetwarzanie wygenerowanego drzewa DOM odbywają się poprzez W3C DOM API.



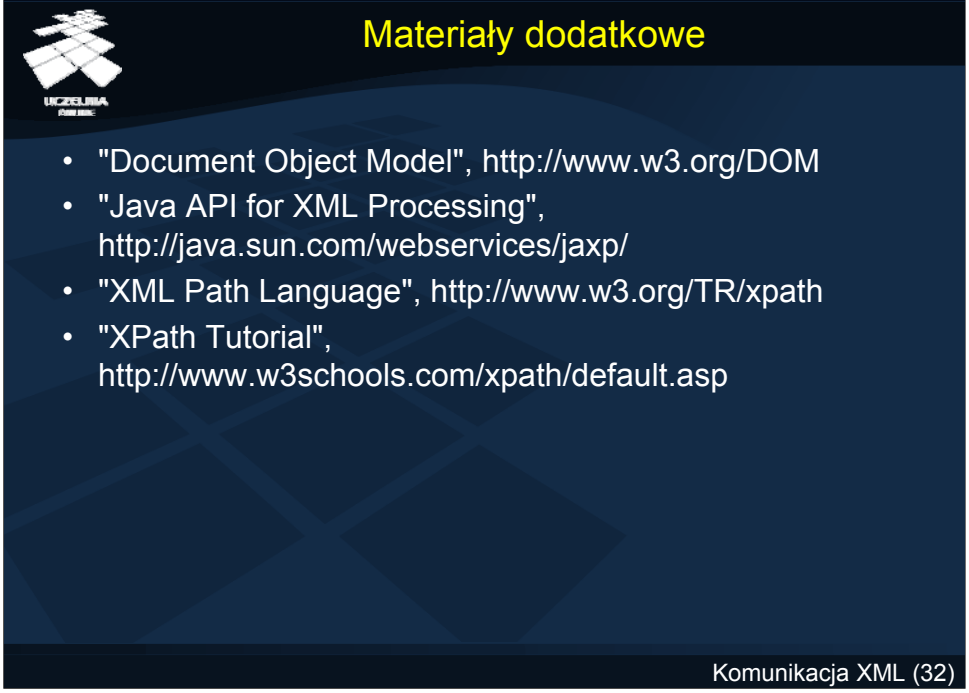
Na slajdzie przedstawiono kod programu Java, który buduje w pamięci operacyjnej strukturę drzewa DOM w oparciu o dokument XML zapisany w pliku "katalog.xml". Następnie w drzewie DOM wyszukiwane są wszystkie węzły o nazwie "tytul", a opisywane przez nie tytuły książek są wyświetlane na ekranie.

Do wywołania parsera DOM służy metoda parse() obiektu klasy DocumentBuilder, który z kolei jest tworzony za pomocą metody newDocumentBuilder() obiektu klasy DocumentBuilderFactory, który wreszcie jest tworzony za pomocą metody newInstance(). Wynikowe drzewo DOM jest reprezentowane przez obiekt interfejsu Document, nazwany w przykładzie "xmlDoc". W dalszej części programu z drzewa DOM wybierane są wszystkie węzły o nazwie "tytul", a pętla wyświetla kolejno wartości ich pierwszych potomków, którymi są brzmienia tytułów kolejnych książek z katalogu.



Model DOM umożliwia programiście oderwanie się od myślenia o danych XML jako o plikach tekstowych wypełnionych dużą liczbą znaczników. Format XML może być traktowany np. wyłącznie jako sposób zapisu danych na dysku lub jako format przesyłania tych danych przez sieć komputerową, natomiast rzeczywista struktura tych danych, podlegająca przetwarzaniu, to łatwa w obsłudze struktura drzewiasta, zwykle implementowana obiektowo. Dzięki opracowaniu standardowej specyfikacji W3C DOM API, dokumenty XML mogą być przetwarzane w taki sam sposób za pomocą dowolnego języka programowania. Programista, który zmienia język programowania, nie musi od nowa poznawać biblioteki obsługi dokumentów XML. Pomimo iż wszystkie przykłady przedstawione podczas wykładu korzystały z języka Java, to jednak ich przeniesienie do innych języków programowania byłoby bardzo łatwe.

W realizacji złożonych operacji przeszukiwania drzew DOM programista może korzystać z języka zapytań XPath, za którego obsługę jest odpowiedzialna biblioteka programistyczna zgodna z W3C DOM API. Z kolei do automatycznej transformacji dokumentów XML do postaci drzew DOM służą moduły parserów DOM.



•"Document Object Model", http://www.w3.org/DOM
•"Java API for XML Processing", http://java.sun.com/webservices/jaxp/
•"XML Path Language", http://www.w3.org/TR/xpath
•"XPath Tutorial", http://www.w3schools.com/xpath/default.asp