

Drzewa DOM

Maciej Zakrzewicz

mzakrz@cs.put.poznan.pl

<http://www.cs.put.poznan.pl/~mzakrz/>

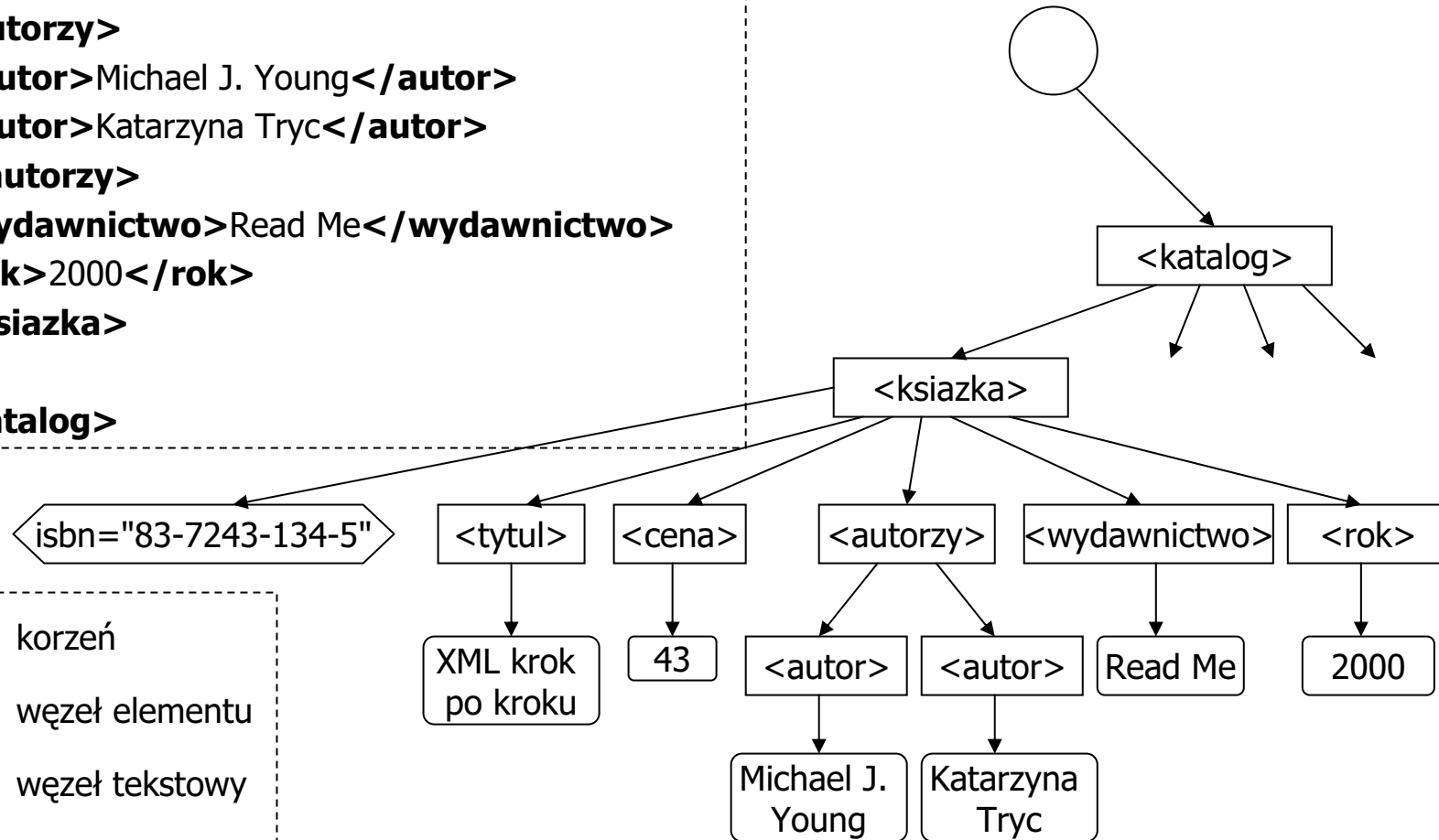
Document Object Model (DOM)

- Document Object Model jest standardem modelowania dokumentów XML przy użyciu struktury drzewa – znaczniki XML i ich zawartość są modelowane przez węzły drzewa; zagnieżdżanie znaczników służy za podstawę do konstruowania hierarchii
- 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 parser DOM
- Implementacja, adresowanie i przeszukiwanie drzew Document Object Model mogą być realizowane przy użyciu biblioteki DOM API

Przykład struktury drzewa DOM

```
<katalog>
  <ksiazka isbn="83-7243-134-5">
    <tytul>XML krok po kroku</tytul>
    <cena>43</cena>
    <autorzy>
      <autor>Michael J. Young</autor>
      <autor>Katarzyna Tryc</autor>
    </autorzy>
    <wydawnictwo>Read Me</wydawnictwo>
    <rok>2000</rok>
  </ksiazka>
  ...
</katalog>
```

- korzeń
- węzeł elementu
- węzeł tekstowy
- ⬡ węzeł atrybutu



W3C DOM API: obiekt Node

- Obiekt klasy/typu **Node** reprezentuje węzeł w drzewie DOM (węzeł elementu, węzeł tekstowy, itd.)

Atrybuty (W3C)

attributes	tablica atrybutów węzła
childNodes	tablica węzłów potomnych
firstChild	pierwszy węzeł potomny
lastChild	ostatni węzeł potomny
nextSibling	prawy węzeł sąsiedni
nodeName	nazwa węzła
nodeType	identyfikator typu węzła
nodeValue	wartość węzła
parentNode	węzeł nadrzędny
previousSibling	lewy węzeł sąsiedni

Metody (W3C)

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

W3C DOM API: obiekt NodeList

- Obiekt klasy/typu **NodeList** reprezentuje zbiór obiektów typu Node

Atrybuty (W3C)

length	liczba elementów w zbiorze
--------	----------------------------

Metody (W3C)

item(i)	zwraca element i-ty element zbioru
---------	------------------------------------

W3C DOM API: obiekt Document

- Obiekt klasy/typu **Document** modeluje całe drzewo DOM; wszystkie węzły drzewa są jego potomkami

Atrybuty (W3C)

documentElement	element najwyższego poziomu w dokumencie
doctype	DTD lub XML Schema dla dokumentu

Metody (W3C)

createAttribute(s)	tworzy nowy węzeł atrybutu
createComment(s)	tworzy nowy węzeł komentarza
createElement(s)	tworzy nowy element
createTextNode(s)	tworzy nowy węzeł tekstowy
getElementsByTagName(s)	zwraca zbiór węzłów o podanej nazwie

W3C DOM API: obiekt Element

- Obiekt klasy/typu **Element** modeluje węzeł reprezentujący znacznik XML

Atrybuty (W3C)

tagName	nazwa węzła
---------	-------------

Metody (W3C)

getAttribute(s)	zwraca wartość podanego atrybutu
getAttributeNode(s)	zwraca węzeł podanego atrybutu
getElementsByTagName(s)	zwraca zbiór węzłów o podanej nazwie
removeAttribute(s)	usuwa wartość podanego atrybutu
removeAttributeNode(n)	usuwa podany węzeł atrybutu
setAttribute(s,s)	ustawia nową wartość atrybutu
setAttributeNode(n)	wstawia nowy węzeł atrybutu

W3C DOM API: obiekt Attr i Text

- Obiekt klasy/typu **Attr** reprezentuje atrybut znacznika XML w formie tzw. węzła atrybutu; obiekt Attr posiada ogólne atrybuty i metody klasy/typu Node plus poniższe:

Atrybuty (W3C)

name	nazwa atrybutu
specified	prawda oznacza, że wartość atrybutu jest ustawiona w dokumencie
value	wartość atrybutu

- Obiekt klasy/typu **Text** reprezentuje treść umieszczoną wewnątrz znacznika XML

Implementacja W3C DOM: Java i PL/SQL

- Wszystkie typy DOM zostały zaimplementowane w języku Java jako interfejsy w pakiecie `org.w3c.dom` (posiadają nazwy jak w specyfikacji W3C) i jako klasy rzeczywiste w pakiecie `oracle.xml.parser.v2` (posiadają nazwy z prefiksem XML)
- Wszystkie typy DOM zostały zaimplementowane w języku PL/SQL jako obiekty pakietu XMLDOM (posiadają nazwy z prefiksem DOM) i DBMS_XMLDOM (od wersji 9.2 DBMS_XMLDOM zaczyna zastępować XMLDOM)

Java: funkcje konstrukcji drzew DOM

- **createElement(String)** [interfejs Document] – tworzy nowy węzeł, reprezentujący znacznik o podanej nazwie; węzeł ten nie wchodzi jeszcze w skład drzewa dokumentu
- **createTextNode(String)** [interfejs Document] – tworzy nowy węzeł tekstowy; węzeł ten nie wchodzi jeszcze w skład drzewa dokumentu
- **appendChild(Node)** [interfejs Node] – dodaje nowy węzeł jako ostatni węzeł potomny
- **cloneNode(boolean)** [interfejs Node] – wykonuje kopię wskazanego węzła wraz z lub bez jego węzłów potomnych
- **removeChild(Node)** [interfejs Node] – odpina wskazany węzeł potomny od jego węzła nadrzędnego
- **replaceChild(Node, Node)** [interfejs Node] – odpina istniejący węzeł potomny i na jego miejscu umieszcza nowy węzeł potomny
- **setNodeValue(String)** [interfejs Node] – nadaje węzłowi wartość tekstową

Konstrukcja drzewa DOM w języku Java

```
XMLDocument xmlDoc = new XMLDocument();
```

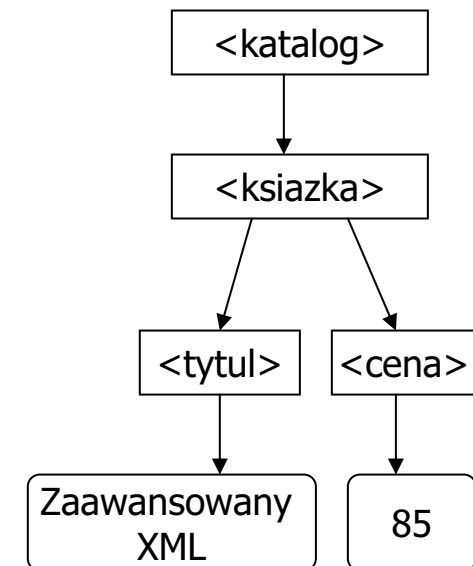
```
Node katalogNode = xmlDoc.createElement("katalog");  
xmlDoc.appendChild(katalogNode);
```

```
Node ksiazkaNode = xmlDoc.createElement("ksiazka");  
katalogNode.appendChild(ksiazkaNode);
```

```
Node tytulNode = xmlDoc.createElement("tytul");  
ksiazkaNode.appendChild(tytulNode);
```

```
Node tytulText = xmlDoc.createTextNode("Zaawansowany XML");  
tytulNode.appendChild(tytulText);
```

```
Node cenaText = xmlDoc.createTextNode("85");  
tytulNode.appendChild(cenaText);
```

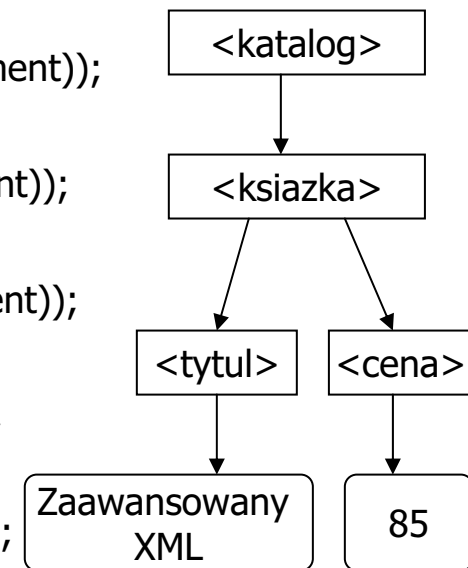


PL/SQL: funkcje konstrukcji drzew DOM

- **xmlDOM.createElement(DOMDocument, Varchar2)** – tworzy nowy węzeł, reprezentujący znacznik o podanej nazwie; węzeł ten nie wchodzi jeszcze w skład drzewa dokumentu
- **xmlDOM.createTextNode(DOMDocument, Varchar2)** – tworzy nowy węzeł tekstowy; węzeł ten nie wchodzi jeszcze w skład drzewa dokumentu
- **xmlDOM.appendChild(DOMDocument, DOMNode)** – dodaje nowy węzeł jako ostatni węzeł potomny
- **xmlDOM.cloneNode(DOMDocument, boolean)** – wykonuje kopię wskazanego węzła wraz z lub bez jego węzłów potomnych
- **xmlDOM.removeChild(DOMDocument, DOMNode)** – odpina wskazany węzeł potomny od jego węzła nadrzędnego
- **xmlDOM.replaceChild(DOMDocument, DOMNode, DOMNode)** – odpina istniejący węzeł potomny i na jego miejscu umieszcza nowy węzeł potomny
- **xmlDOM.setNodeValue(DOMDocument, Varchar2)** – nadaje węzłowi wartość tekstową
- **xmlDOM.makeNode(DOMElement)** – konwertuje typ DOMElement do DOMNode

Konstrukcja drzewa DOM w języku PL/SQL

```
declare
xmlDoc xmldom.DOMDocument; tmpNode xmldom.DOMNode;
katalogElement xmldom.DOMELEMENT; ksiazkaElement xmldom.DOMELEMENT;
tytulElement xmldom.DOMELEMENT; cenaElement xmldom.DOMELEMENT;
cenaText xmldom.DOMText; tytulText xmldom.DOMText;
tytulNode xmldom.DOMNode; cenaNode xmldom.DOMNode;
begin
xmlDoc := xmldom.newDOMDocument;
katalogElement := xmldom.createElement(xmlDoc, 'katalog');
  tmpNode := xmldom.appendChild(xmlDoc, xmldom.makeNode(katalogElement));
ksiazkaElement := xmldom.createElement(xmlDoc, 'ksiazka');
  tmpnode := xmldom.appendChild(tmpNode, xmldom.makeNode(ksiazkaElement));
tytulElement := xmldom.createElement(xmlDoc, 'tytul');
  tytulNode := xmldom.appendChild(tmpNode, xmldom.makeNode(tytulElement));
cenaElement := xmldom.createElement(xmlDoc, 'cena');
  cenaNode := xmldom.appendChild(tmpNode, xmldom.makeNode(cenaElement));
tytulText := xmldom.createTextNode(xmlDoc, 'Zaawansowany XML');
  tmpnode := xmldom.appendChild(tytulNode, xmldom.makeNode(tytulText));
cenaText := xmldom.createTextNode(xmlDoc, '85');
  tmpnode := xmldom.appendChild(cenaNode, xmldom.makeNode(cenaText));
end;
```



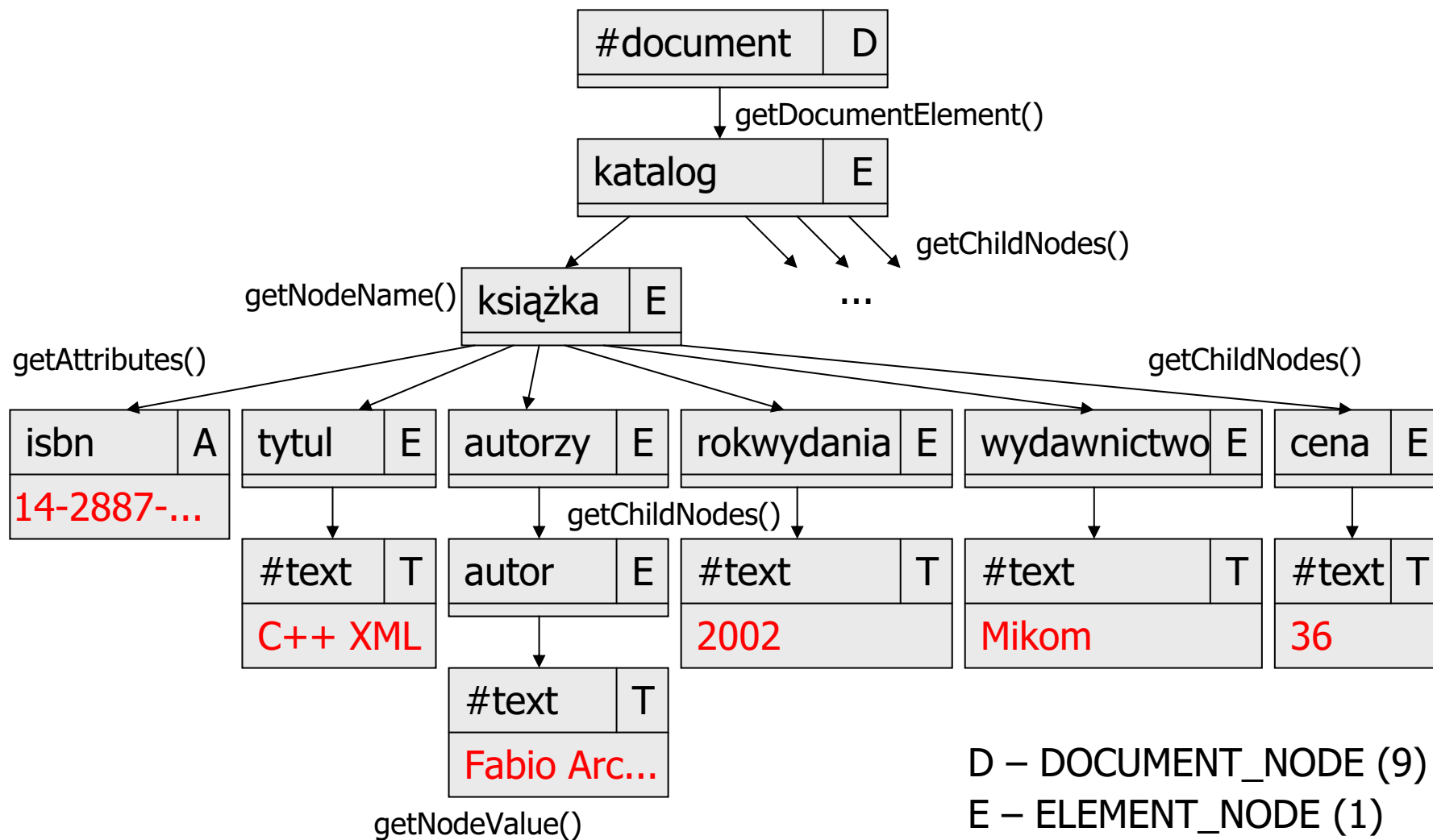
Java: funkcje nawigacyjne DOM API

- **getDocumentElement()** [interfejs Document] – zwraca obiekt węzła reprezentującego znacznik najwyższego poziomu
- **getElementsByTagName(String)** [interfejs Document] – zwraca tablicę obiektów węzłów reprezentujących podany znacznik XML
- **getChildNodes()** [interfejs Node] – zwraca tablicę obiektów węzłów potomnych (bez węzłów atrybutowych)
- **getAttributes()** [interfejs Node] – zwraca tablicę obiektów potomnych węzłów atrybutowych
- **getNodeName()** [interfejs Node] – zwraca nazwę znacznika dla węzła
- **getNodeType()** [interfejs Node] – zwraca numeryczny identyfikator typu węzła
- **getNodeValue()** [interfejs Node] – zwraca treść węzła (tylko dla węzłów tekstowych)
- **getFirstChild()** [interfejs Node] - zwraca obiekt pierwszego węzła potomnego (z pominięciem węzłów atrybutowych)
- **getLastChild()** [interfejs Node] - zwraca obiekt ostatniego węzła potomnego (z pominięciem węzłów atrybutowych)
- **getNextSibling()** [interfejs Node] – zwraca obiekt prawego sąsiada węzła (z pominięciem węzłów atrybutowych)
- **getPreviousSibling()** [interfejs Node] – zwraca obiekt lewego sąsiada węzła (z pominięciem węzłów atrybutowych)
- **getParentNode()** [interfejs Node] – zwraca obiekt węzła nadrzędnego

PL/SQL: funkcje nawigacyjne DOM API

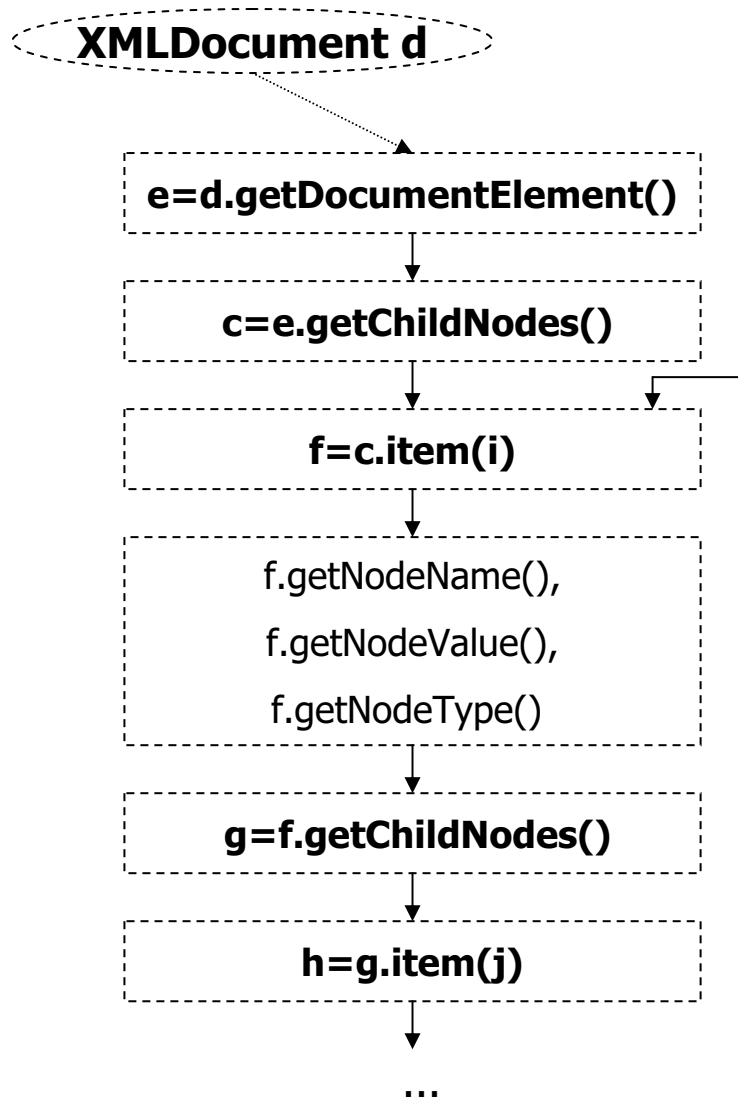
- **xmlDOM.getDocumentElement(DOMDocument)** – zwraca obiekt węzła reprezentującego znacznik najwyższego poziomu
- **xmlDOM.getElementsByTagName(DOMDocument)** – zwraca tablicę obiektów węzłów reprezentujących podany znacznik XML
- **xmlDOM.getChildNodes(DOMNode)** – zwraca tablicę obiektów węzłów potomnych (bez węzłów atrybutowych)
- **xmlDOM.getAttributes(DOMNode)** – zwraca tablicę obiektów potomnych węzłów atrybutowych
- **xmlDOM.getNodeName(DOMNode)** – zwraca nazwę znacznika dla węzła
- **xmlDOM.getNodeType(DOMNode)** – zwraca numeryczny identyfikator typu węzła
- **xmlDOM.getNodeValue(DOMNode)** – zwraca treść węzła (tylko dla węzłów tekstowych)
- **xmlDOM.getFirstChild(DOMNode)** - zwraca obiekt pierwszego węzła potomnego (z pominięciem węzłów atrybutowych)
- **xmlDOM.getLastChild(DOMNode)** - zwraca obiekt ostatniego węzła potomnego (z pominięciem węzłów atrybutowych)
- **xmlDOM.getNextSibling(DOMNode)** – zwraca obiekt prawego sąsiada węzła (z pominięciem węzłów atrybutowych)
- **xmlDOM.getPreviousSibling(DOMNode)** – zwraca obiekt lewego sąsiada węzła (z pominięciem węzłów atrybutowych)
- **xmlDOM.getParentNode(DOMNode)** – zwraca obiekt węzła nadrzędnego

DOM API: funkcje nawigacyjne



D – DOCUMENT_NODE (9)
 E – ELEMENT_NODE (1)
 A – ATTRIBUTE_NODE (2)
 T – TEXT_NODE (3)

Java: prosta nawigacja w drzewie DOM



Odczytaj węzeł reprezentujący znacznik najwyższego poziomu (np. <katalog>)

Pobierz listę elementów potomnych

W pętli odczytuj kolejne węzły z listy (np. <ksiazka>)

Przetwarzaj węzeł

Pobierz listę elementów potomnych

Odczytaj kolejny węzeł z listy (np. <tytul>, <autorzy>, <cena>, <rok>, <wydawnictwo>)

Java: nawigacja w drzewie DOM

`getChildNodes()`

Wyświetl tytuły wszystkich książek opisanych w dokumencie XML

```
XMLDocument xmlDoc;
...
Node docNode = null, bookNode = null, elementNode = null;
NodeList docNodeList = null, bookNodeList = null;
try {
    docNode = xmlDoc.getDocumentElement();
    docNodeList = docNode.getChildNodes();
    for (int i=0; i<docNodeList.getLength(); i++) {
        bookNode = docNodeList.item(i);
        bookNodeList = bookNode.getChildNodes();
        for (int j=0; j<bookNodeList.getLength(); j++) {
            elementNode = bookNodeList.item(j);
            if (elementNode.getNodeName().equals("tytul"))
                System.out.println(elementNode.getFirstChild().getNodeValue());
        }
    }
} catch (Exception e) {System.out.println(e);}
}
```

```
Access 2002. Projektowanie baz
danych. Księga eksperta
Access 2002/XP PL dla każdego
ASP.NET. Vademecum profesjonalisty
C++ XML
Dane w sieci WWW
Delphi 6. Praktyka programowania -
tom 1,2
Delphi. Almanach
...
```

Java: nawigacja w drzewie DOM

`getAttributes()`

Wyświetl wartość pierwszego atrybutu każdego znacznika `<ksiazka>` w dokumencie XML

```
XMLDocument xmlDoc;
```

```
...
```

```
Node docNode = null, bookNode = null, urlNode = null;
```

```
NodeList docNodeList = null, bookNodeList = null;
```

```
try {
```

```
    docNode = xmlDoc.getDocumentElement();
```

```
    docNodeList = docNode.getChildNodes();
```

```
    for (int i=0; i<docNodeList.getLength(); i++) {
```

```
        bookNode = docNodeList.item(i);
```

```
        urlNode = bookNode.getAttributes().item(0);
```

```
        System.out.println(urlNode.getNodeValue());
```

```
    }
```

```
} catch (Exception e) {System.out.println(e);} }
```

```
83-7197-669-0
```

```
83-7197-786-7
```

```
83-7197-691-7
```

```
83-7279-215-1
```

```
83-7279-149-X
```

```
83-7279-214-3
```

```
83-7197-469-8
```

```
83-7197-377-2
```

```
...
```

Java: nawigacja w drzewie DOM

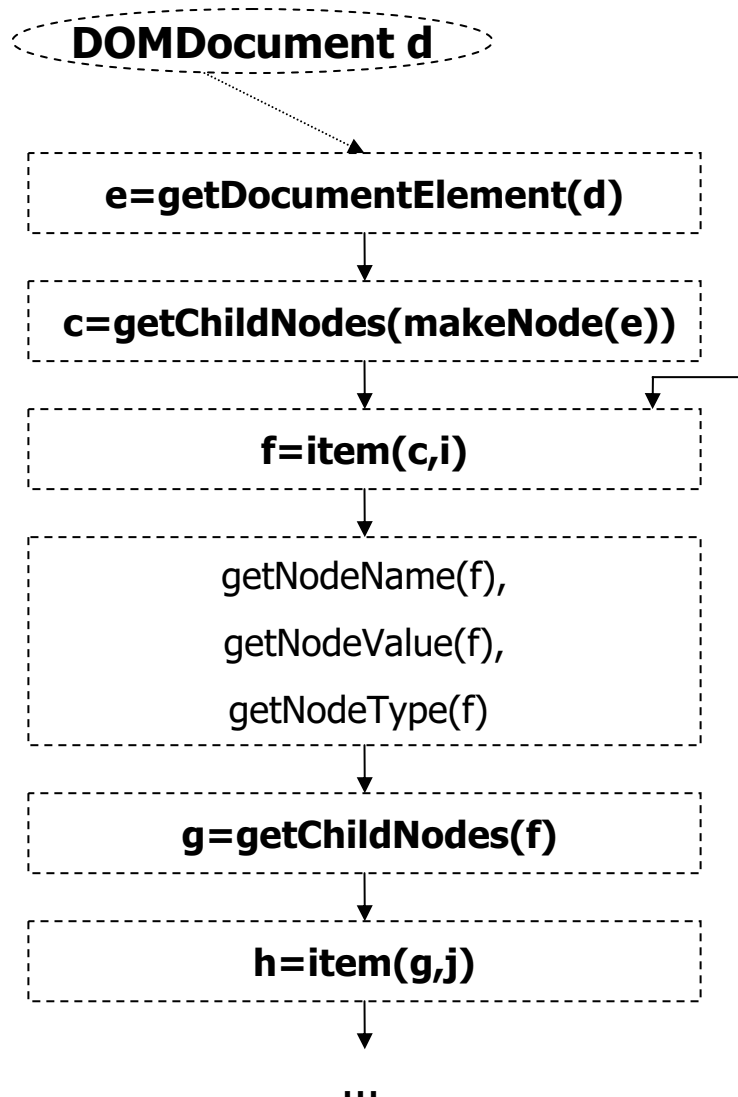
`getElementsByTagName()`

Wyświetl tytuły wszystkich książek opisanych w dokumencie XML

```
XMLDocument xmlDoc;  
...  
Node titleNode = null;  
NodeList titleNodeList = null;  
try {  
    titleNodeList = xmlDoc.getElementsByTagName("tytul");  
    for (int i=0; i<titleNodeList.getLength(); i++) {  
        titleNode = titleNodeList.item(i);  
        System.out.println(titleNode.getFirstChild().getNodeValue();)  
    }  
} catch (Exception e) {System.out.println(e);}
```

```
Access 2002. Projektowanie baz  
danych. Księga eksperta  
Access 2002/XP PL dla każdego  
ASP.NET. Vademecum profesjonalisty  
C++ XML  
Dane w sieci WWW  
Delphi 6. Praktyka programowania -  
tom 1,2  
Delphi. Almanach  
...
```

PL/SQL: prosta nawigacja w drzewie DOM



Odczytaj węzeł reprezentujący znacznik najwyższego poziomu (np. <katalog>)

Pobierz listę elementów potomnych

W pętli odczytaj kolejne węzły z listy (np. <ksiazka>)

Przetwarzaj węzeł

Pobierz listę elementów potomnych

Odczytaj kolejny węzeł z listy (np. <tytul>, <autorzy>, <cena>, <rok>, <wydawnictwo>)

PL/SQL: nawigacja w drzewie DOM

getChildNodes()

Wyświetl tytuły wszystkich książek opisanych w dokumencie XML

```
declare
xmlDoc xmldom.DOMDocument; xmlElem xmldom.DOMELEMENT; bookNode xmldom.DOMNode;
bookNodeList xmldom.DOMNodeList; innerNode xmldom.DOMNode; innerNodeList xmldom.DOMNodeList;
begin
...
xmlElem := xmldom.getDocumentElement(xmlDoc);
bookNodeList := xmldom.getChildNodes(xmldom.makeNode(xmlElem));
for i in 0..xmldom.getLength(bookNodeList) - 1 loop
    bookNode := xmldom.item(bookNodeList, i);
    innerNodeList := xmldom.getChildNodes(bookNode);
    for j in 0..xmldom.getLength(innerNodeList) - 1 loop
        innerNode := xmldom.item(innerNodeList, j);
        if (xmldom.getNodeName(innerNode) = 'tytul') then
            dbms_output.put_line(xmldom.getNodeValue(xmldom.getFirstChild(innerNode)));
        end if;
    end loop;
end loop;
xmldom.freeDocument(xmlDoc);
...
end;
```

```
Access 2002. Projektowanie baz danych.
Księga eksperta
Access 2002/XP PL dla każdego
ASP.NET. Vademecum profesjonalisty
C++ XML
Dane w sieci WWW
Delphi 6. Praktyka programowania - tom
1,2
Delphi. Almanach
...
```

PL/SQL: nawigacja w drzewie DOM

`getAttributes()`

Wyświetl wartość pierwszego atrybutu każdego znacznika `<ksiazka>` w dokumencie XML

```
declare
xmlDoc xmldom.DOMDocument; xmlElem xmldom.DOMELEMENT;
bookNode xmldom.DOMNode; bookNodeList xmldom.DOMNodeList;
attrNode xmldom.DOMNode;
begin
...
xmlElem := xmldom.getDocumentElement(xmlDoc);
bookNodeList := xmldom.getChildNodes(xmldom.makeNode(xmlElem));
for i in 0..xmldom.getLength(bookNodeList) - 1 loop
    bookNode := xmldom.item(bookNodeList, i);
    attrNode := xmldom.item(xmldom.getAttributes(bookNode),0);
    dbms_output.put_line(xmldom.getNodeValue(attrNode));
end loop;
xmldom.freeDocument(xmlDoc);
...
end;
```

```
83-7197-669-0
83-7197-786-7
83-7197-691-7
83-7279-215-1
83-7279-149-X
83-7279-214-3
83-7197-469-8
83-7197-377-2
...
```

PL/SQL: nawigacja w drzewie DOM

`getElementsByTagName()`

Wyświetl tytuły wszystkich książek opisanych w dokumencie XML

```
declare
xmlDoc xmldom.DOMDocument;
titleNode xmldom.DOMNode;
titleNodeList xmldom.DOMNodeList;
begin
...
titleNodeList := xmldom.getElementsByTagName(xmlDoc,'tytul');
for i in 0..xmldom.getLength(titleNodeList) - 1 loop
  titleNode := xmldom.item(titleNodeList, i);
  dbms_output.put_line(xmldom.getNodeValue(xmldom.getFirstChild(titleNode)));
end loop;
xmldom.freeDocument(xmlDoc);
...
end;
```

```
Access 2002. Projektowanie baz
danych. Księga eksperta
Access 2002/XP PL dla każdego
ASP.NET. Vademecum profesjonalisty
C++ XML
Dane w sieci WWW
Delphi 6. Praktyka programowania -
tom 1,2
Delphi. Almanach
...
```

Java: konwersja drzewa DOM do pliku XML

```
...  
try {  
    xmlDoc.print(new FileOutputStream("C:\\katalog.xml"));  
} catch (Exception e) {System.out.println(e);}  
...
```

```
<?xml version = '1.0' encoding = 'WINDOWS-1250'?>  
<katalog>  
  <ksiazka url="http://dot.com/19821.html">  
    <tytul>C++ XML</tytul>  
    <autorzy>  
      <autor>Fabio Arciniegas</autor>  
    </autorzy>  
    <rokwydania>2002</rokwydania>  
    <wydawnictwo>Mikom</wydawnictwo>  
    <cena>36</cena>  
  </ksiazka>  
  <ksiazka url="http://dot.com/19854.html">  
  ...
```

PL/SQL: konwersja drzewa DOM do pliku XML

...

```
xmlDom.writeToFile(xmlDoc,/mydir/a.xml');
```

...

```
<?xml version = '1.0' encoding = 'WINDOWS-1250'?>  
<katalog>  
  <ksiazka url="http://dot.com/19821.html">  
    <tytul>C++ XML</tytul>  
    <autorzy>  
      <autor>Fabio Arciniegas</autor>  
    </autorzy>  
    <rokwydania>2002</rokwydania>  
    <wydawnictwo>Mikom</wydawnictwo>  
    <cena>36</cena>  
  </ksiazka>  
  <ksiazka url="http://dot.com/19854.html">  
  ...
```

Język XPath

Maciej Zakrzewicz

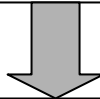
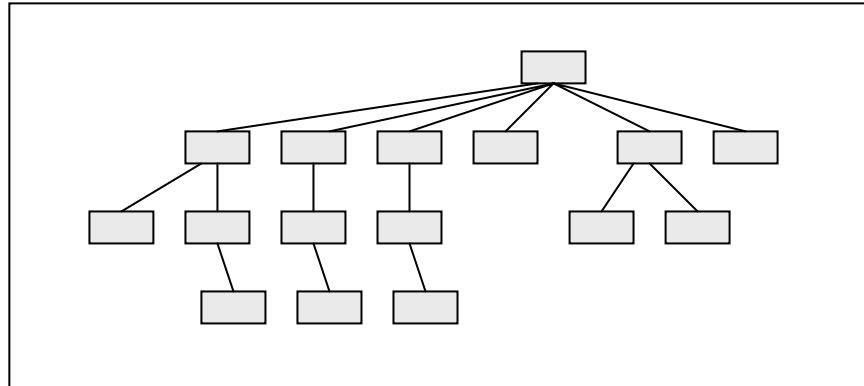
mzakrz@cs.put.poznan.pl

<http://www.cs.put.poznan.pl/~mzakrz/>

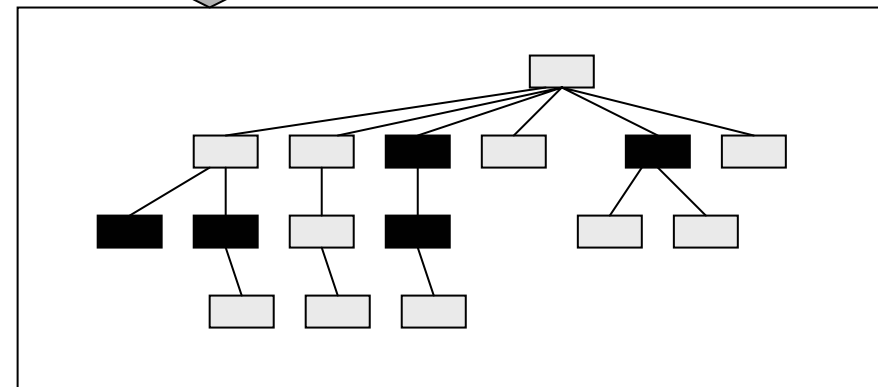
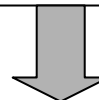
Język XPath

- XPath to specyfikacja języka służącego do adresowania, odczytywania i przeszukiwania drzew DOM dokumentów XML
- XPath odgrywa podobną rolę w stosunku do drzew DOM, jak język SQL w stosunku do relacyjnych baz danych
- XPath stosuje notację przypominającą ścieżki dostępu w systemach plików
- Wynikiem ewaluacji wyrażenia XPath jest zbiór węzłów spełniających warunki selekcji
- XPath pozwala stosować dwa rodzaje zapisu wyrażień: skrócony i pełny; rodzaje te mogą być mieszane

Przetwarzanie wyrażeń XPath



Wyrażenie XPath



Skrócone wyrażenia XPath (1/3)

- Wybór węzłów w drzewie DOM:
 - wybierz autorów wszystkich książek
`/katalog/ksiazka/autorzy`
 - wybierz wszystkie wydawnictwa dowolnie zagłębione w drzewie
`//wydawnictwo`
 - wybierz wszystkie węzły potomne (dzieci) każdego węzła książka
`//ksiazka/*`
- Wybór n-tego węzła danego rodzaju:
 - wybierz pierwszego autora każdej książki
`//autorzy/autor[1]`
 - wybierz drugiego autora pierwszej książki
`//ksiazka[1]//autor[2]`
 - wybierz ostatnią książkę
`//ksiazka[last()]`

Skrócone wyrażenia XPath (2/3)

- Wybór węzłów, które posiadają podany węzeł potomny:

- wybierz książki, które posiadają autorów

```
//książka[autorzy]
```

- wybierz książki, które napisał Serge Abiteboul

```
//książka[autorzy/autor="Serge Abiteboul"]
```

- Alternatywa ścieżek:

- wybierz tytuły książek i wydawnictwa

```
//tytuł | //wydawnictwo
```

- wybierz tytuły książek napisanych przez Serge'a Abiteboula lub Kurta Walla

```
//tytuł[../autor="Serge Abiteboul"] | //tytuł[../autor="Kurt Wall"]
```

Skrócone wyrażenia XPath (3/3)

- Wybór węzłów zawierających atrybuty:

- wybierz książkę o numerze ISBN "83-7279-149-X"

```
//książka[@isbn="83-7279-149-X"]
```

- wybierz wszystkie numery ISBN

```
//@isbn
```

- wybierz książki, które posiadają numer ISBN

```
//książka[@isbn]
```

- Odczyt treści węzła:

- odczytaj treści wszystkich tytułów książek

```
//książka/tytuł/text()
```

Pełne wyrażenia XPath

- Wyrażenie ścieżkowe XPath składa się z tzw. kroków rozdzielonych ukośnikami
- W pełnym zapisie, każdy z kroków może składać się z:
 - **specyfikatora współrzędnych** (axis), służącego do określenia miejsca w drzewie, począwszy od którego wyszukiwane będą węzły
 - **testu węzła** (node test), służącego do określenia, które węzły są wyszukiwane w obszarze drzewa określonym przez specyfikator współrzędnych
 - **predykatów**, dodatkowo zawężających test węzła, powodujących wybór tylko tych węzłów, które spełniają podany warunek
- Każdy krok pełnego wyrażenia XPath zapisywany jest przy użyciu następującej notacji:
spec_współrzędnych::test_węzła[predykaty]

Specyfikatory współrzędnych

ancestor	obejmuje wszystkie węzły nadrzędne (ojciec, dziadek, itd.) bieżącego węzła
ancestor-or-self	obejmuje bieżący węzeł plus wszystkie węzły nadrzędne
attribute	obejmuje wszystkie atrybuty bieżącego węzła
child	obejmuje wszystkie węzły bezpośrednio podrzędne bieżącego węzła (dzieci)
descendant	obejmuje wszystkie węzły podrzędne (syn, wnuk, itd.) bieżącego węzła
descendant-or-self	obejmuje bieżący węzeł plus wszystkie węzły podrzędne
following	obejmuje wszystkie węzły, które w dokumencie następują za węzłem bieżącym
following-sibling	obejmuje wszystkie węzły sąsiednie, które w dokumencie następują za węzłem bieżącym
parent	obejmuje węzeł bezpośrednio nadrzędny bieżącego węzła (ojciec)
preceding	obejmuje wszystkie węzły, które w dokumencie następują przed węzłem bieżącym
preceding-sibling	obejmuje wszystkie węzły sąsiednie, które w dokumencie następują przed węzłem bieżącym
self	obejmuje bieżący węzeł

Testy węzłów, operatory i funkcje

node()	dowolny węzeł
text()	węzeł tekstowy
*	dowolny element
@*	dowolny atrybut
<i>nazwa</i>	węzeł o podanej nazwie

=, !=, <, >, <=, >=	porównania
or, and	operatory logiczne
+, -, *, div, mod	operatory arytmetyczne
count()	liczba węzłów wybieranych przez wyrażenie
last()	liczba porządkowa ostatniego węzła wybieranego przez wyrażenie
name()	nazwa węzła wybieranego przez wyrażenie
position()	liczba porządkowa węzła wybieranego przez wyrażenie
not()	negacja logiczna
true()	zwraca prawdę logiczną
false()	zwraca fałsz logiczny

Funkcje - ciąg dalszy

Funkcje operujące na tekstach

concat()	konkatenacja tekstów
contains()	test zawierania tekstów
starts-with()	sprawdzenie, czy tekst rozpoczyna się od podanego ciągu
string()	konwersja do tekstu
string-length()	długość tekstu
substring()	ekstrakcja podciągu znaków
substring-after()	zwraca ciąg znaków znajdujący się za wycinanym podciągiem
substring-before()	zwraca ciąg znaków znajdujący się przed wycinanym podciągiem
translate()	dokonyje zamiany wszystkich wystąpień podanego podciągu

Funkcje operujące na liczbach

ceiling()	górne domknięcie całkowite
floor()	dolne domknięcie całkowite
number()	konwersja do liczby
round()	zaokrąglenie
sum()	suma zbioru wartości liczbowych

Przykłady wyrażeń XPath (1/2)

- Wybierz książki, których cena nie przekracza 20 zł
//książka[cena<=20]
- Wybierz tytuły książek o cenach w przedziale 30-40 zł
//książka/tytuł[../cena>30 and ../cena<40]
- Wybierz książki napisane przez więcej niż dwóch autorów
//książka[count(autorzy/autor)>2]
- Wybierz co drugą książkę
//książka[position() mod 2 = 1]
- Wybierz książki zawierające w tytule słowo XML
//książka[contains(tytuł,"XML")]
- Wybierz autorów o imieniu "Stephen"
//autor[starts-with(., "Stephen")]
- Wybierz tytuły złożone z ponad 20 znaków
//tytuł[string-length(.)>20]

Przykłady wyrażeń XPath (2/2)

- Wybierz książki, których numer ISBN spełnia wzorzec *****7197*******

//książka[substring(@isbn,4,4) = "7197"]

- Wybierz wszystkie węzły autorzy oraz wszystkie ich węzły potomne

//autorzy/descendant-or-self::*

- Wybierz wszystkie książki, które w dokumencie znajdują się za książką o numerze ISBN "83-7197-786-7"

//książka[@isbn="83-7197-786-7"]/following::książka

- Wybierz wszystkie atrybuty pierwszej książki

//książka[1]/attribute::*

- Wybierz cenę książki pt. "XML dla każdego"

//książka[tytuł="XML dla każdego"]/child::cena

Transformacja wyrażeń skróconych do pełnych

skrót	pełne	przykład
brak	child::	//ksiazka/cena => //ksiazka/child::cena
@	attribute::	//katalog/ksiazka[@isbn= "83-7197-786-7"] => //katalog/child::ksiazka[attribute::isbn= "83-7197-786-7"]
.	self::node()	//tytul[string-length(.)>10] => //tytul[string-length(self::node())>10]
..	parent::node()	//autor/.. => //autor/parent::node()
//	/descendant-or-self::node()	//tytul => /descendant-or-self::node()

Funkcje XPath w DOM API

Java

- **selectNodes(String)** [interfejs Node] – zwraca tablicę obiektów węzłów spełniających podaną ścieżkę XPath
- **selectSingleNode(String)** [interfejs Node] – zwraca pierwszy znaleziony obiekt węzła spełniającego podaną ścieżkę XPath
- **valueOf(String)** [interfejs Node] – zwraca treść pierwszego znalezionej obiektu węzła spełniającego podaną ścieżkę XPath

PL/SQL

- **xslprocessor.selectNodes(DOMNode, Varchar2)** – zwraca tablicę obiektów węzłów spełniających podaną ścieżkę XPath
- **xslprocessor.selectSingleNode(DOMNode, Varchar2)** – zwraca pierwszy znaleziony obiekt węzła spełniającego podaną ścieżkę XPath
- **xslprocessor.valueOf(DOMNode, Varchar2)** – zwraca treść pierwszego znalezionej obiektu węzła spełniającego podaną ścieżkę XPath

Java: zapytania XPath

Wyświetl tytuły wszystkich książek wydanych w roku 2002

```
XMLDocument xmlDoc;  
...  
Node titleNode = null;  
NodeList queryNodeList = null;
```

```
try {  
    queryNodeList = xmlDoc.selectNodes("//ksiazka[rokwydania='2002']/tytul");  
    for (int i=0; i<queryNodeList.getLength(); i++) {  
        titleNode = queryNodeList.item(i);  
        System.out.println(titleNode.getFirstChild().getNodeValue();)  
    }  
} catch (Exception e) {System.out.println(e);}
```

C++ XML

Flash i XML. Techniki zaawansowane

HTML and XML dla początkujących

Programowanie Microsoft SQL Server 2000 z XML

Vademecum XML

XML Kompendium programisty

Java: zapytania XPath

Wyświetl tytuły wszystkich książek, których jeden z autorów ma imię "Fabio"

C++ XML

XML Kompendium programisty

```
XMLDocument xmlDoc;  
...  
Node titleNode = null;  
NodeList queryNodeList = null;  
  
try {  
    queryNodeList =  
        xmlDoc.selectNodes("//tytul[../autorzy[contains(autor,'Fabio')]]");  
    for (int i=0; i<queryNodeList.getLength(); i++) {  
        titleNode = queryNodeList.item(i);  
        System.out.println(titleNode.getFirstChild().getNodeValue());  
    }  
} catch (Exception e) {System.out.println(e);}
```

Java: zapytania XPath

Wyświetl nazwisko pierwszego autora książki pt. "Java i XML"

```
Brett McLaughlin
```

```
try {  
    System.out.println(xmlDoc.valueOf("//ksiazka[tytul='Java i XML']//autor"));  
} catch (Exception e) {System.out.println(e);} }
```

PL/SQL: zapytania XPath

Wyświetl tytuły wszystkich książek wydanych w roku 2002

```
declare
xmlDoc xmldom.DOMDocument;
titleNode xmldom.DOMNode;
queryNodeList xmldom.DOMNodeList;
begin
...
queryNodeList := xslprocessor.selectNodes(xmlDom.makeNode(xmlDoc),
'//ksiazka[rokwydania="2002"]/tytul');
for i in 0..xmldom.getLength(queryNodeList) - 1 loop
titleNode := xmldom.item(queryNodeList, i);
dbms_output.put_line(xmldom.getNodeValue(xmldom.getFirstChild(titleNode)));
end loop;
xmldom.freeDocument(xmlDoc);
...
end;
```

```
C++ XML
Flash i XML. Techniki zaawansowane
HTML and XML dla początkujących
Programowanie Microsoft SQL Server 2000 z XML
Vademecum XML
XML Kompendium programisty
```

PL/SQL: zapytania XPath

Wyświetl tytuły wszystkich książek, których jeden z autorów ma imię "Fabio"

```
declare
xmlDoc xmldom.DOMDocument;
titleNode xmldom.DOMNode;
queryNodeList xmldom.DOMNodeList;
begin
...
queryNodeList := xslprocessor.selectNodes(xmlDom.makeNode(xmlDoc),
'//tytul[../autorzy[contains(autor,"Fabio")]]');
for i in 0..xmldom.getLength(queryNodeList) - 1 loop
titleNode := xmldom.item(queryNodeList, i);
dbms_output.put_line(xmldom.getNodeValue(xmldom.getFirstChild(titleNode)));
end loop;
xmldom.freeDocument(xmlDoc);
...
end;
```

C++ XML

XML Kompendium programisty

PL/SQL: zapytania XPath

Wyświetl nazwisko pierwszego autora książki pt. "Java i XML"

Brett McLaughlin

```
declare
xmlDoc xmlDom.DOMDocument;
begin
...
dbms_output.put_line( xslprocessor.valueOf(xmlDom.makeNode(xmlDoc),
'//ksiazka[tytul="Java i XML"]//autor');
...
end;
```