Programowanie współbieżne i rozproszone

WYKŁAD 7

dr inż. Krzysztof Pancerz

XML

- XML (ang. eXtensible Markup Language - rozszerzalny język znaczników) definiuje ogólną składnię, stosowaną przy oznaczaniu danych za pomocą znaczników.
- XML oferuje standardowy format dokumentów.
- XML może służyć do opisu praktycznie wszystkich rodzajów danych.

Wybrane standardy związane z językiem XML

- **DTD** (Definicje typu dokumentu) - opis struktury (typu) dokumentów.
- **XML Schema** - schematy zawartości dokumentów XML.
- **XSLT** (Transformacje XSL) - transformacje dokumentów XML do innych języków i formatów (np. HTML) albo do innych dokumentów XML.
- **XPath** - wyszukiwanie i wskazywanie na fragmenty dokumentów XML.

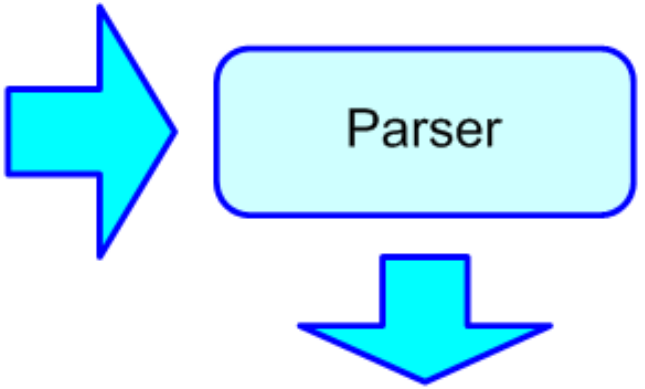
Wybrane standardy związane z językiem XML (cd.)

- **XLink** - łączenie zasobów XML.
- **DOM** (obiektowy model dokumentu) - przetwarzanie dokumentów XML traktowanych jak drzewa obiektów (węzłów).

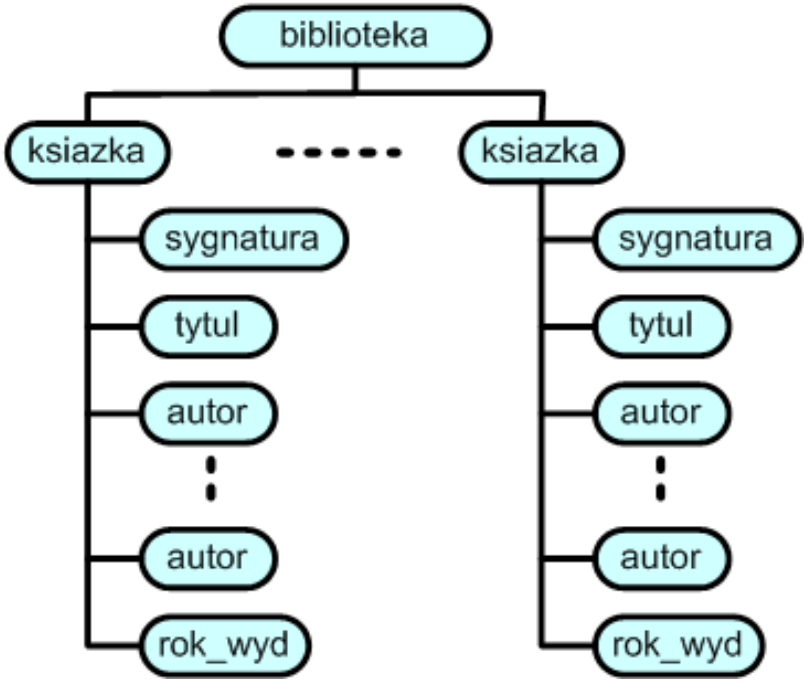
Przetwarzanie dokumentów XML

Dokument XML

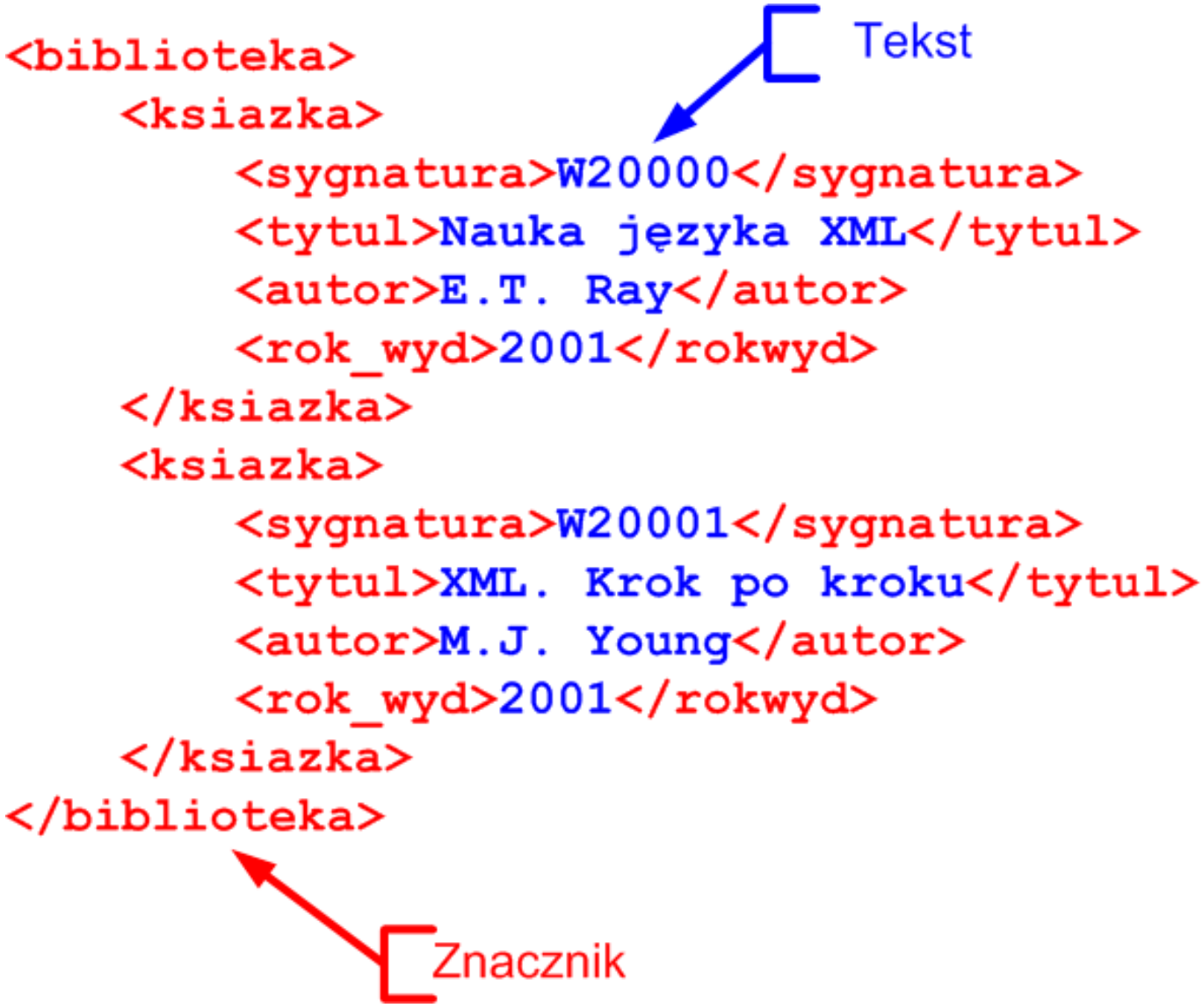
```
<biblioteka>
<ksiazka>
<sygnatura>W20000</sygnatura>
<tytul>Nauka języka XML</tytul>
<autor>E.T. Ray</autor>
<rok_wyd>2001</rokwyd>
</ksiazka>
<ksiazka>
<sygnatura>W20001</sygnatura>
<tytul>XML. Krok po kroku</tytul>
<autor>M.J. Young</autor>
<rok_wyd>2001</rokwyd>
</ksiazka>
</biblioteka>
```



Reprezentacja wewnętrzna



Dokument XML (cd.)



Nazwy znaczników

- XML nie posiada ustalonego zbioru znaczników w przeciwieństwie do języka HTML, gdzie nazwy znaczników są predefiniowane.
- XML jest **metajęzykiem** i dostarcza możliwość definiowania własnych znaczników. Jednak narzuca on pewną gramatykę, która kontroluje rozmieszczanie znaczników, miejsca ich pojawienia się, dozwolone nazwy, sposoby wiązania atrybutów z elementami, itp.

Znaczniki

- Znaczniki występujące w dokumencie XML opisują tylko strukturę oraz semantykę dokumentu.
- Znaczniki nie informują o sposobie prezentacji (wyświetlania) dokumentu.

Elementy

- Elementy dzielą dokument na części składowe.
- Elementy mogą być pojemnikami zawierającymi tekst lub inne elementy lub mogą być puste.
- Dla elementów pustych wartość informacyjną ma tylko ich położenie i atrybuty.

Elementy (cd.)


<ocena>
<punkty>
<kolokwium> 20 </kolokwium>
<obecnosc> 5 </obecnosc>
<aktywnosc/>
</punkty>
</ocena>

Dane znakowe

- Wszystko co nie jest elementem jest tekstem (danymi znakowymi).
- Specyfikacja XML zawiera predefiniowane jednostki znakowe, które pozwalają posługiwać się znakami specjalnymi. Istnieje też możliwość wykorzystania numerycznych jednostek znakowych, które mają postać &#n, gdzie n jest kodem znaku w Unicode.

Prolog dokumentu

- Na początku dokumentu XML znajduje się opcjonalna informacja specjalna, nazywana prologiem dokumentu.
- W pierwszej linii prologu znajduje się deklaracja XML. Następnie w prologu mogą pojawić się definicja typu dokumentu oraz instrukcje przetwarzania.
- Po prologu umieszczony jest element bazowy dokumentu.

Prolog dokumentu (cd.)

```
<?xml version="1.0"?>
```

Prolog dokumentu

```
<biblioteka>
    <ksiazka>
        ...
    </ksiazka>
    ...
    <ksiazka>
        ...
    </ksiazka>
</biblioteka>
```

Deklaracja XML

- Deklaracja XML opisuje niektóre najogólniejsze cechy dokumentu:
  - numer wersji (**version**),
  - kodowanie znaków używanych w dokumencie (**encoding**),
  - informacja czy dokument jest samodzielnym dokumentem XML (**standalone**).

Deklaracja XML (cd.)

```
<?xml
    version="1.0"?
    encoding="ISO-8859-1"
    standalone="yes"
?>
```

Przestrzenie nazw

- Przestrzenie nazw pozawalają zapewnić jednoznaczność nazw i zapobiec konfliktom nazw zasobów pochodzących z różnych źródeł.
- W dokumentach XML przestrzenie nazw umożliwiają bezkolizyjne stosowanie w jednym dokumencie elementów pochodzących z różnych języków zdefiniowanych w oparciu o XML.

Przestrzenie nazw

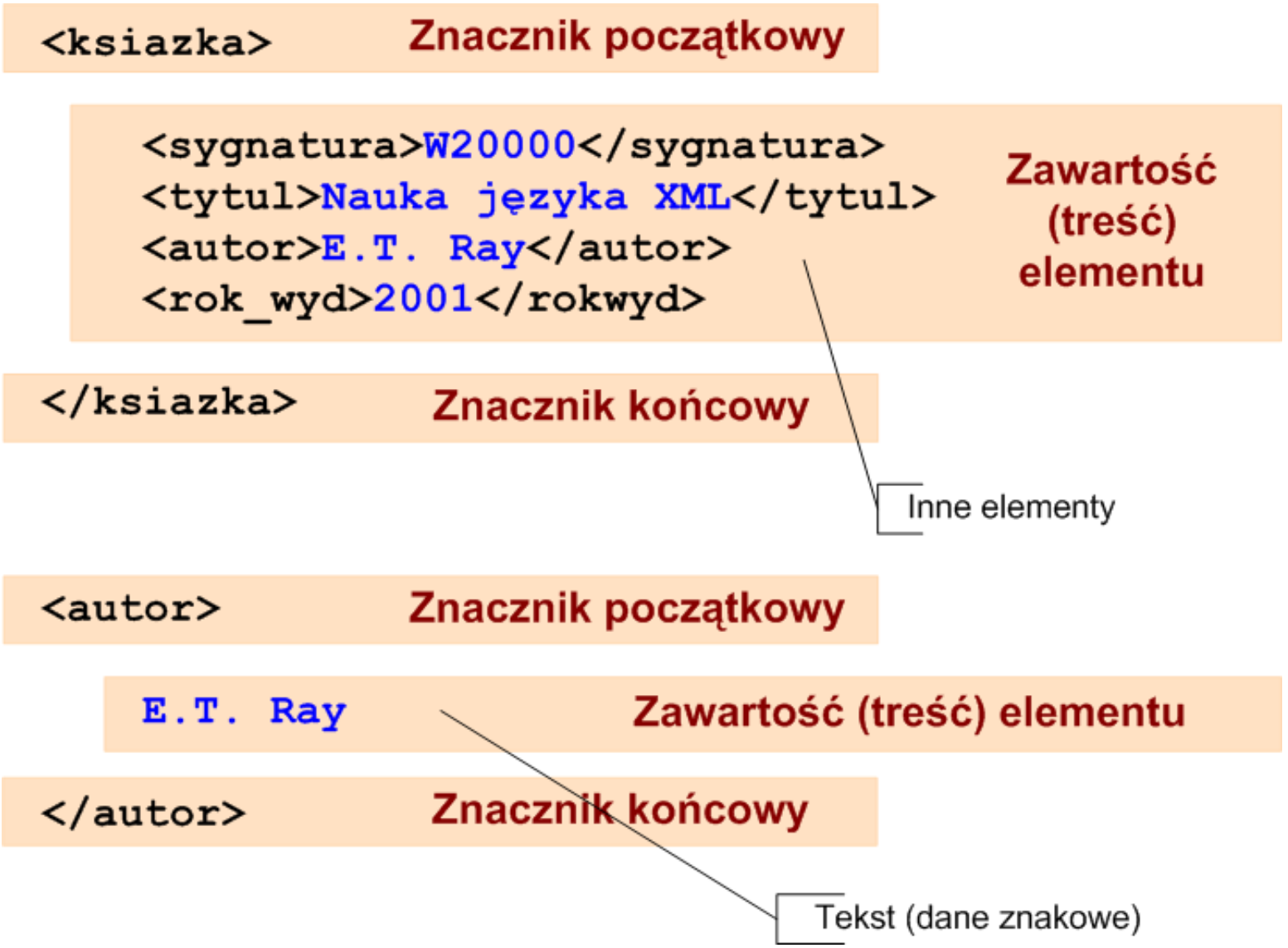
Przykład:

```
<element
xmlns:prefiks=”adres url prefiksu”>
...
<prefiks:nazwa ...>
</element>
```

Element pojemnika

- Element pojemnika zawiera inne elementy lub tekst (dane znakowe). Zwartość może też być mieszana (inne elementy, dane znakowe).

Element pojemnika (cd.)



Element pusty

- Element pusty pozbawiony jest zawartości (treści).

```
<ksiazka/>
```

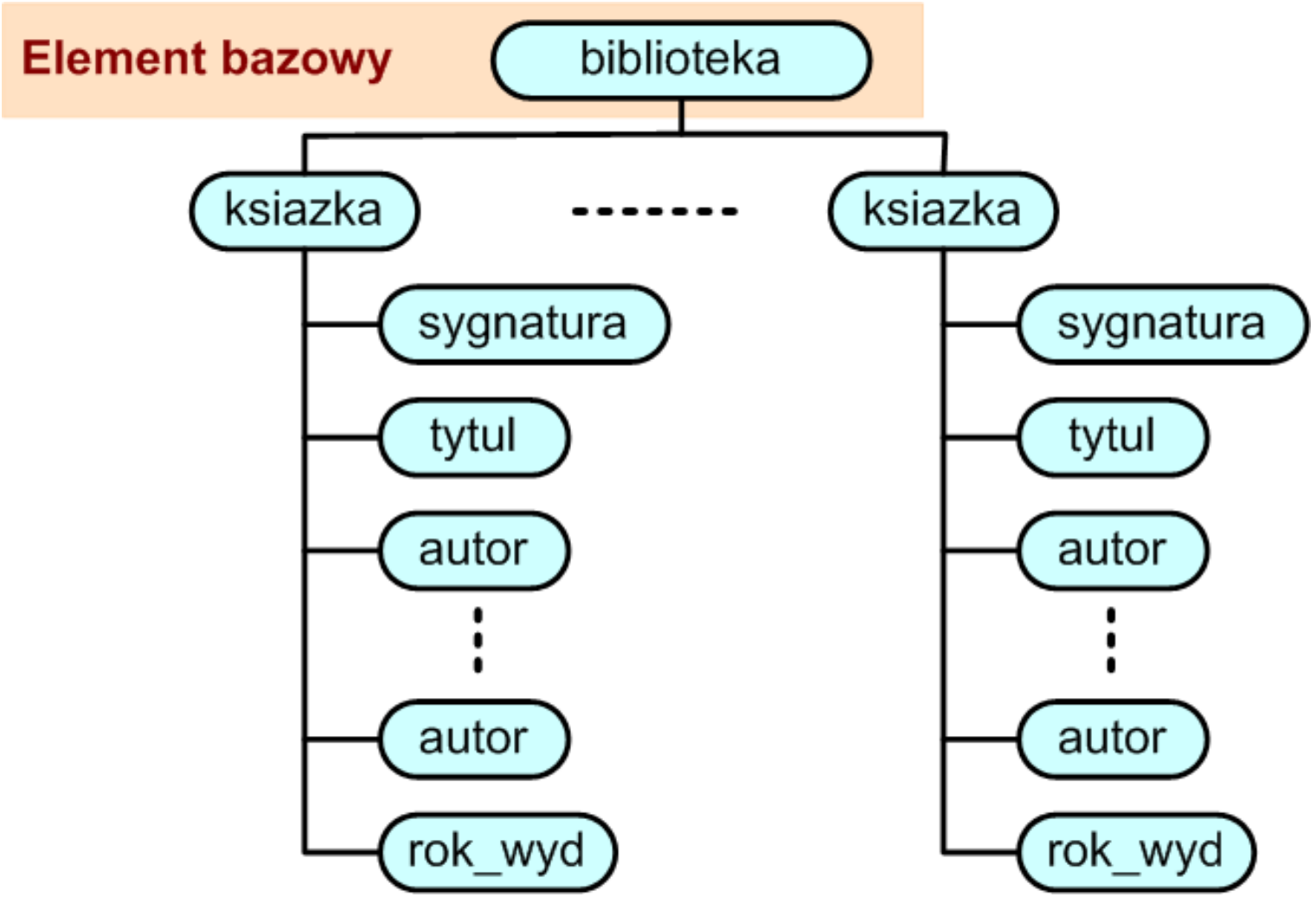
Znacznik

```
<ksiazka/>  ≡  <ksiazka></ksiazka>
```

Struktura drzewa

- Dokumenty XML posiadają hierarchiczną strukturę drzewa, w której jedne elementy są całkowicie zagnieżdżone w innych.
- Każdy dokument XML zawiera dokładnie jeden element bazowy (element główny, element dokumentu) nie posiadający rodzica, który zawiera wszystkie pozostałe elementy.
- Element taki wyznacza granicę dokumentu.

Struktura drzewa (cd.)



Struktura drzewa (cd.)

```
<biblioteka>  Element bazowy
    <ksiazka>
        <sygnatura>W20000</sygnatura>
        <tytul>Nauka języka XML</tytul>
        <autor>E.T. Ray</autor>
        <rok_wyd>2001</rokwyd>
    </ksiazka>
    <ksiazka>
        <sygnatura>W20001</sygnatura>
        <tytul>XML. Krok po kroku</tytul>
        <autor>M.J. Young</autor>
        <rok_wyd>2001</rokwyd>
    </ksiazka>
</biblioteka>
```

Atrybuty

- Elementy mogą zawierać tzw. atrybuty, które dostarczają dodatkowych informacji o elemencie.
- Element może mieć dowolną liczbę atrybutów, pod warunkiem, że każdy atrybut posiada niepowtarzalną nazwę.
- Atrybuty dołączane są do znacznika początkowego (dla elementu - pojemnika) lub do znacznika elementu pustego i są parami *nazwa-wartość*.

Atrybuty (cd.)

```
nazwa="wartość"
```

Atrybut

```
<ksiazka sygnatura="W20000">

    <tytul>Nauka języka XML</tytul>
    <autor>E.T. Ray</autor>
    <rok_wyd>2001</rokwyd>

</ksiazka>
```

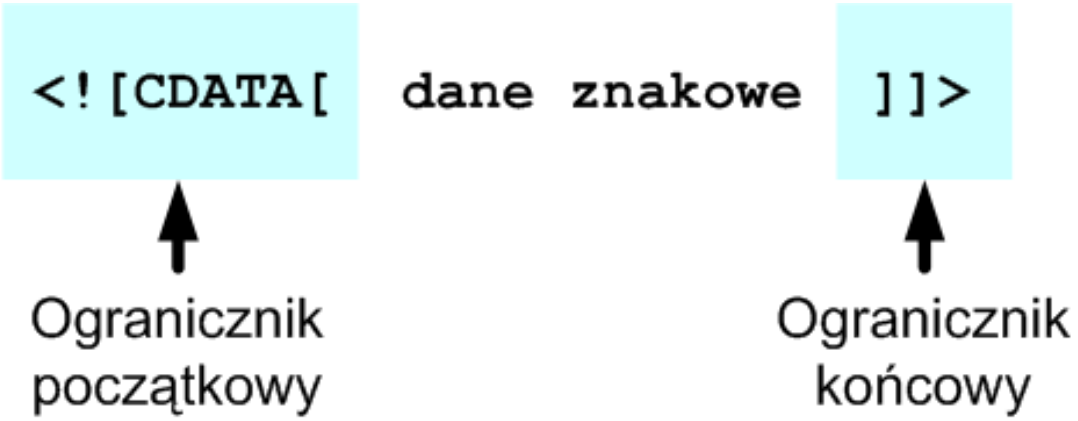
Atrybuty

```
<autor imie="Eric" nazwisko="Ray"/>
```

Sekcje CDATA

- Sekcje CDATA pozwalają na wstawianie do dokumentu XML tekstu zawierającego symbole zarezerwowane (jak np. < i &).
- Sekcję CDATA rozpoczyna się ciągiem znaków <![CDATA[oraz kończy ciągiem znaków]]>.
- Wszystko co znajduję się pomiędzy tymi ogranicznikami jest traktowane jako pierwotne dane znakowe.

Sekcje CDATA (cd.)




```
<rownanie numer=”20">
    <![CDATA[
    Jeśli a>=b wtedy c=10
    ]]>
</rownanie>
```

Sekcja CDATA

Konstrukcja dokumentów XML

- Zasady poprawnej konstrukcji dokumentu XML:
  - każdy znacznik początkowy musi posiadać odpowiadający mu znacznik końcowy,
  - elementy nie mogą wzajemnie zachodzić na siebie,
  - może występować dokładnie jeden element bazowy,
  - każda wartość atrybutu musi być ujęta w cudzysłów,
  - żaden element nie może mieć dwóch atrybutów o tej samej nazwie,
  - komentarze i instrukcje przetwarzania nie mogą występować wewnątrz znaczników.

Modele przetwarzania dokumentów XML w języku Java

- Model SAX
- Model DOM
- Model JDOM

Model SAX

- Model SAX (*Simple API for XML*) definiuje sposób przetwarzania dokumentów XML.
- SAX nie jest parserem (tzn. sam nie dokonuje przetwarzania dokumentów XML).
- SAX określa sposób odczytywania dokumentów XML i rozbijania danych na odpowiednie elementy za pomocą mechanizmu obsługi zdarzeń.
- SAX definiuje zdarzenia procesu przetwarzania dokumentu XML, które podlegają monitorowaniu i obsłudze.

Model SAX (cd.)

- Parser działający w oparciu o model SAX przetwarza dokument XML i za każdym razem, gdy napotyka jakiś znacznik, komentarz, dane tekstowe lub jakikolwiek inny element dokumentu sygnalizuje odpowiednie zdarzenie.
- W obsłudze sygnalizowanego przez parser zdarzenia mogą zostać wykonane odpowiednie działania.

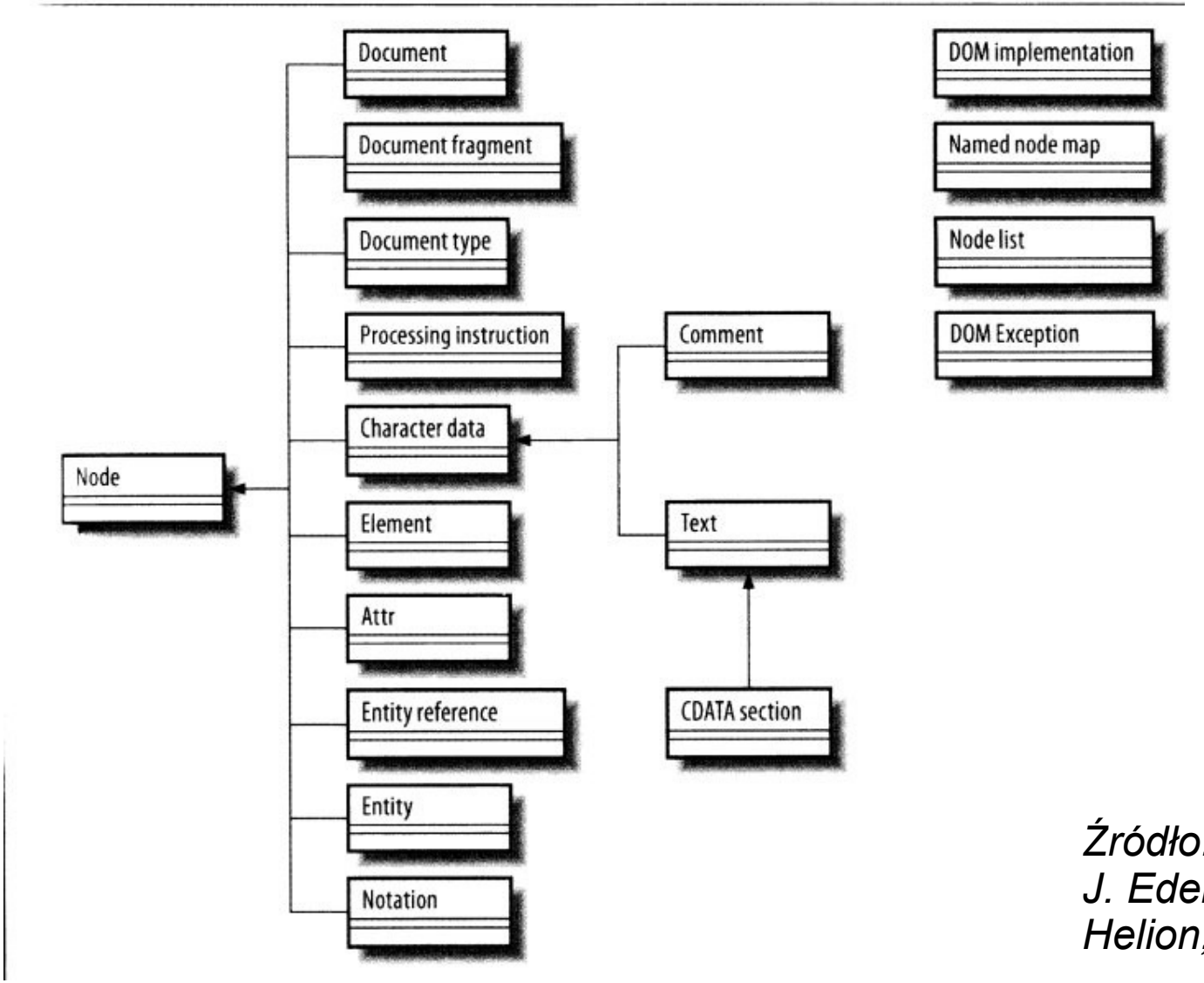
Model DOM

- Model DOM (*Document Object Model*) został zdefiniowany przez W3C (*World Wide Web Consortium*) DOM Working Group.
- DOM umożliwia dostęp do danych oraz manipulowanie danymi w dokumentach XML.
- Dokument XML w modelu DOM reprezentowany jest za pomocą struktury drzewa.
- Cały dokument XML wczytywany jest do pamięci, a wszystkie dane umieszczane są w węzłach drzewa.

Model DOM (cd.)

- DOM jest specyfikacją niezależną od platformy i języka programowania.
- Model DOM jest dostępny praktycznie we wszystkich językach programowania.

Hierarchia interfejsów modelu DOM w języku Java związanych z reprezentacją dokumentu XML



Źródło: B.D. McLaughlin, J. Edelson: Java i XML. Helion, Gliwice, 2007.

Podstawowe klasy i interfejsy modelu DOM

DocumentBuilderFactory

- klasa abstrakcyjna
- “fabryka” API umożliwiająca otrzymanie parsera tworzącego obiekt DOM z dokumentu XML
- wybrane metody:
  - **newInstance()** – tworzy instancję „fabryki”
  - **newDocumentBuilder()** – tworzy nową instancję DocumentBuilder

Podstawowe klasy i interfejsy modelu DOM (cd.)

DocumentBuilder

- klasa abstrakcyjna
- definicja API umożliwiającego otrzymanie obiektu DOM z dokumentu XML
- wybrana metoda:
  - **parse(File f)** – parsuje dokument XML z podanego pliku i zwraca obiekt DOM

Podstawowe klasy i interfejsy modelu DOM (cd.)

Document

- interfejs
- wewnętrzna reprezentacja dokumentu XML
- wybrane metody:
  - **createElement(String tagName)** – tworzy w dokumencie element o podanej nazwie
  - **getDocumentElement()** – zwraca element dokumentu

Podstawowe klasy i interfejsy modelu DOM (cd.)

Element

- interfejs
- element dokumentu XML
- wybrane metody:
  - **setAttribute(String name, String value)** – ustawia wartość podanego atrybutu
  - **getAttribute(String name)** – zwraca wartość podanego atrybutu
  - **getElementsByTagName(String name)** – zwraca listę węzłów potomnych o podanej nazwie

Podstawowe klasy i interfejsy modelu DOM (cd.)

Node

- interfejs
- węzeł dokumentu XML
- wybrane metody:
  - **appendChild(Node newChild)** – dodaje do węzła podany węzeł potomny
  - **getChildNodes()** - pobiera wszystkie węzły potomne danego węzła

Podstawowe klasy i interfejsy modelu DOM (cd.)

NodeList

- interfejs
- lista węzłów dokumentu XML
- wybrane metody:
  - **getLength()** – zwraca rozmiar listy

Porównanie modeli SAX i DOM

Pobieranie danych:

SAX: dane pobierane w obsłudze zdarzeń

DOM: dane pobierane ze struktury drzewa

Dostęp do danych:

SAX: dostęp sekwencyjny do danych

DOM: dostęp swobodny do danych

Porównanie modeli SAX i DOM (cd.)

Zużycie pamięci:

SAX: małe zużycie pamięci, możliwe przetwarzanie w pamięci części dokumentu XML

DOM: znaczne zużycie pamięci, w pamięci przetwarzany jest cały dokument XML

Przetwarzanie:

SAX: przetwarzanie jednokrotne dokumentu

DOM: przetwarzanie wielokrotne dokumentu

JDOM

- JDOM umożliwia dostęp do dokumentów XML z poziomu języka Java poprzez strukturę drzewa.
- JDOM został zaprojektowany specjalnie dla języka Java dzięki czemu jest bardziej intuicyjny od DOM.
- JDOM składa się z konkretnych klas umożliwiających bezpośrednie tworzenie obiektów.
- W JDOM używane są kolekcje języka Java, np. kolekcja *List* zamiast listy *NodeList* modelu DOM.

Pakiety dla języka Java

SAX

- org.xml.sax
- org.xml.sax.helpers, org.xml.sax.ext

DOM

- org.w3c.dom

JDOM

- org.jdom
- org.jdom.input, org.jdom.output, org.jdom.adapters, org.jdom.filter, org.jdom.transform

Protokół SOAP

- Na bazie XML zdefiniowano protokoły wykorzystywane w aplikacjach rozproszonych i sieciowych, np. protokół SOAP.
- SOAP - Simple Object Access Protocol (Prosty protokół dostępu do obiektów) - protokół wykorzystywany do wywoływania metod serwerów, serwisów, komponentów oraz obiektów).
- Dane w postaci XML przesyłane są zasadniczo za pomocą protokołu HTTP.

Protokół SOAP

- Specyfikacja SOAP określa rodzaje nagłówków HTTP oraz definiuje słownik znaczników XML wykorzystywanych do reprezentacji parametrów, zwracanych wartości oraz generowanych przez metody wyjątków.

Protokół SOAP

- Struktura komunikatu SOAP:
  - **Envelope** - opakowanie, element bazowy (korzeń dokumentu).
  - **Header** - nagłówek (opcjonalny). Bloki nagłówka (podelementy elementu Header) zawierają informacje precyzujące sposób traktowania wiadomości (np. kodowanie).
  - **Body** - ciało zawierające parametry wywołania / odpowiedź / informacje o błędach.

Protokół SOAP

- Cechy komunikatu SOAP:
  - musi być zapisany w formacie XML
  - musi zawierać element Envelope
  - może zawierać element Header
  - musi zawierać element Body
  - musi używać odpowiednich przestrzeni nazw
  - nie może zawierać odwołania do DTD
  - nie może zawierać instrukcji przetwarzania XML

Protokół SOAP

Przykład struktury komunikatu SOAP:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

Źródło: http://www.w3schools.com/

Protokół SOAP

Przykład komunikatu żądania SOAP:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

Źródło: http://www.w3schools.com/

Protokół SOAP

Przykład komunikatu odpowiedzi SOAP:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

Źródło: http://www.w3schools.com/