# Metodyka tworzenia portali biznesowych

## WYKŁAD 5

# Formaty wymiany danych w aplikacjach biznesowych

- XML (ang. *eXtensible Markup Language*) rozszerzalny język znaczników.
- JSON (ang. *JavaScript Object Notation*).

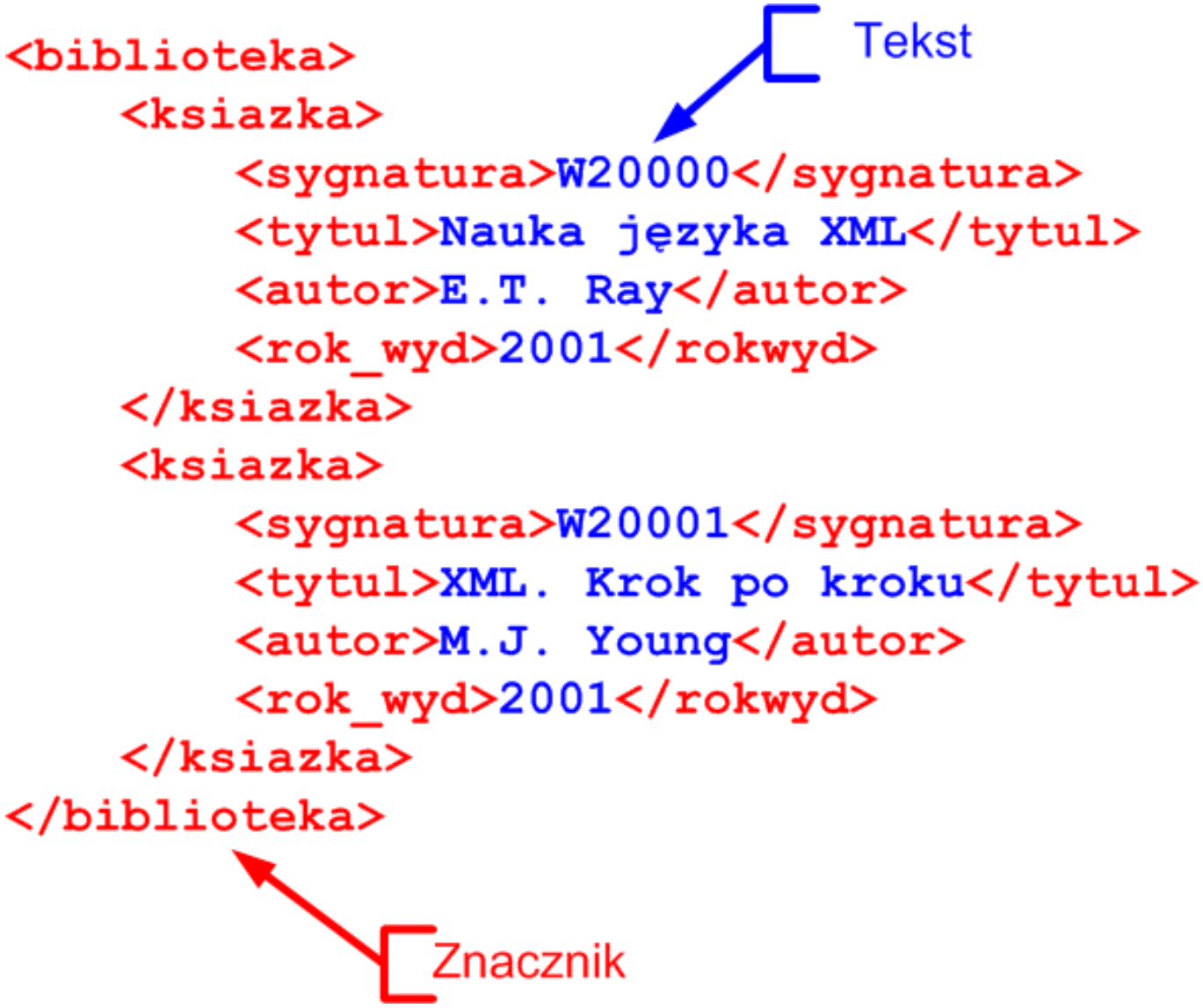
# XML

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

# Dokument XML

- Dokument XML, z rozszerzeniem .xml, jest plikiem tekstowym zawierającym tzw. znaczniki oraz dane tekstowe.
- Znaczniki rozgraniczają i etykietują części dokumentu oraz dodają inne informacje, które pomagają w zdefiniowaniu struktury. Tekst między znacznikami jest treścią dokumentu.
- Dzięki temu, że zawarta w dokumentach XML informacja jest oznakowana, łatwe jest jej znalezienie, pobranie, sortowanie, filtrowanie, porządkowanie, przetwarzanie, itp.

# 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).
- Niektóre znaki nie mogą być używane w tekście dokumentu XML ponieważ posiadają specjalne przeznaczenie (np. nawiasy trójkątne <>).
- 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 specjalna informacja, 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.)

- Dokumenty mogą nie zawierać deklaracji XML. Jeśli jednak dokument posiada taką deklarację musi ona być umieszczona w pierwszej linii prologu.

# 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>
```

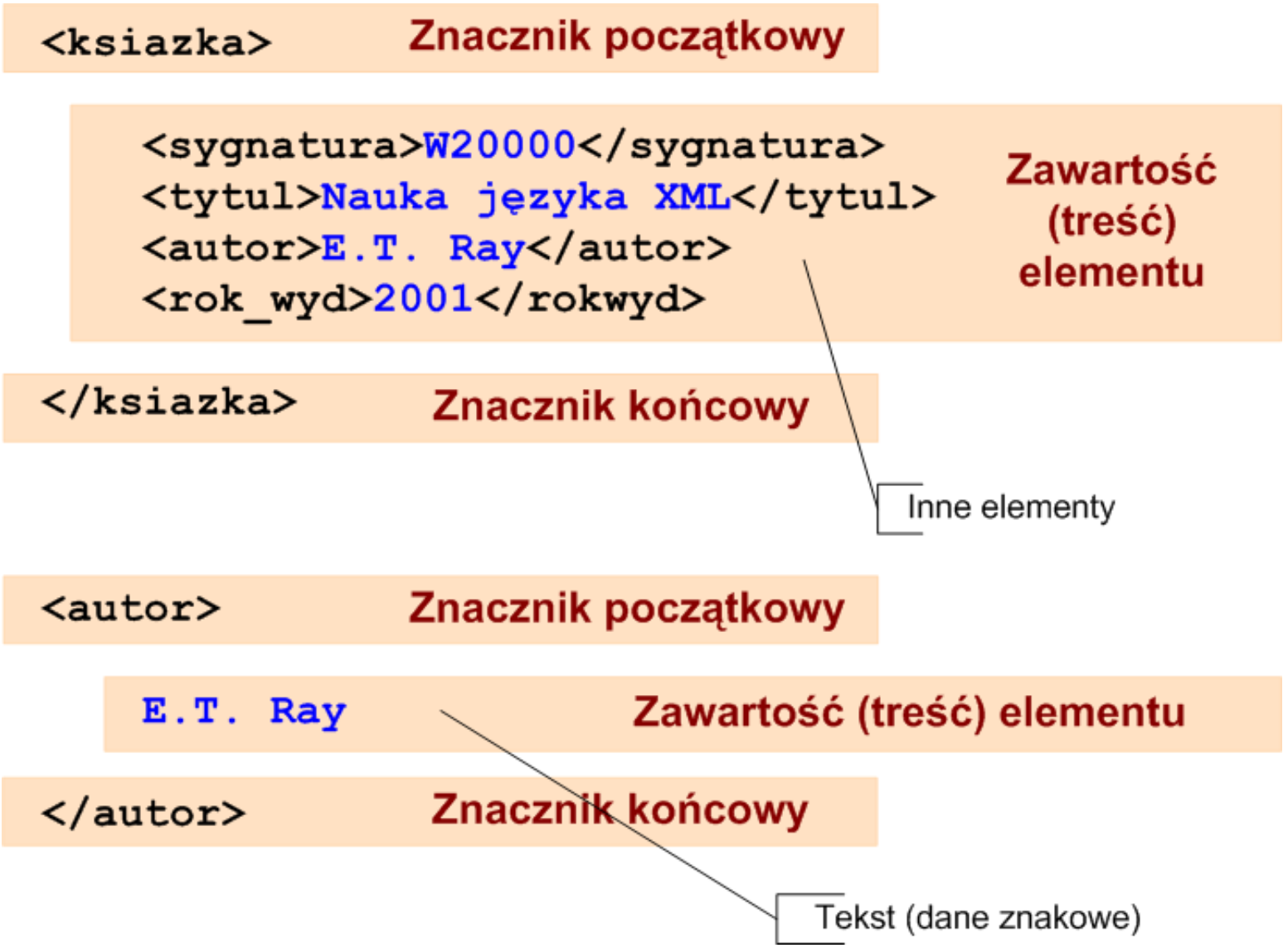
# Budowa elementu - pojemnika

- Element pojemnika zaczyna się od znacznika początkowego, składającego się z otwierającego nawiasu trójkątnego (<), nazwy oraz zamykającego nawiasu trójkątnego (>).
- Znacznik początkowy może ponadto zawierać po nazwie pewne atrybuty oddzielone znakami odstępu.
- Po znaczniku początkowym pojawia się zawartość (treść) elementu, a po treści - znacznik końcowy.

# Budowa elementu – pojemnika (cd.)

- Znacznik końcowy składa się z otwierającego nawiasu trójkątnego (<), ukośnika (/), nazwy elementu i nawiasu zamykającego (>).
- Zawartością elementu mogą być inne elementy lub tekst (dane znakowe). Zwartość może też być mieszana (inne elementy, dane znakowe).

# Budowa elementu – pojemnika (cd.)



# Budowa elementu pustego

- Element pusty pozbawiony jest zawartości (treści) i składa się z jednego znacznika, który zaczyna się od otwierającego nawiasu trójkątnego (<), po którym następuje nazwa elementu.
- Po nazwie mogą wystąpić atrybuty.
- Znacznik kończy się ukośnikiem (/) i zamykającym nawiasem trójkątnym (>).

# Budowa elementu pustego (cd.)

```
<ksiazka/>
```

Znacznik

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

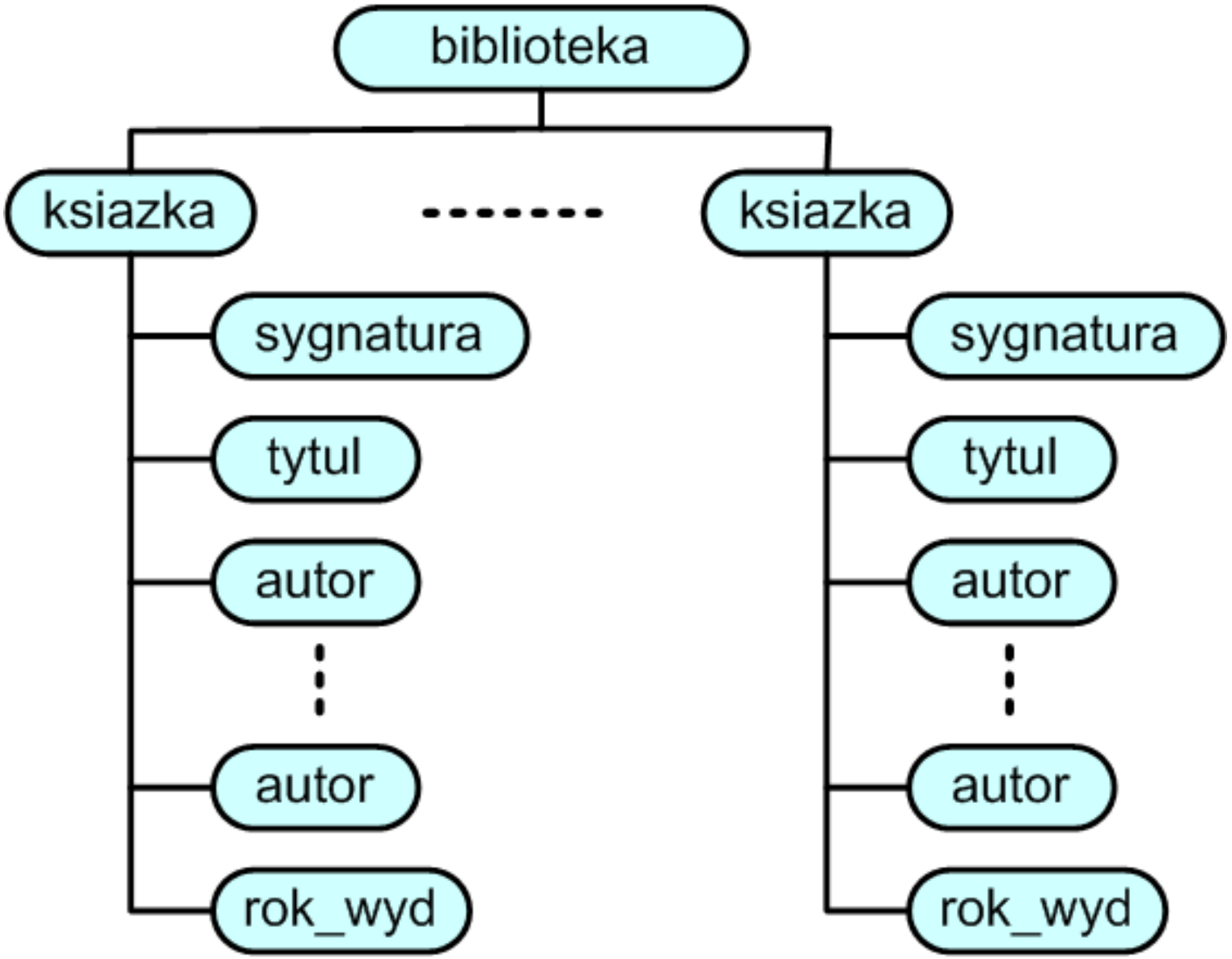
# Struktura drzewa

- Dokumenty XML posiadają hierarchiczną strukturę drzewa, w której jedne elementy są całkowicie zagnieżdżone w innych.
- Element zawierający inne elementy (nazywane elementami potomnymi) jest elementem rodzicielskim.
- Każdy element (z wyjątkiem elementu głównego) posiada dokładnie jeden element rodzicielski.
- Każdy element może posiadać wiele elementów potomnych.

# Struktura drzewa (cd.)

- Elementy mogą być zagnieżdżane, ale tylko wewnętrznie (jeden element całkowicie wewnątrz drugiego). Wzajemne zachodzenie elementów na siebie jest w XML zabronione.

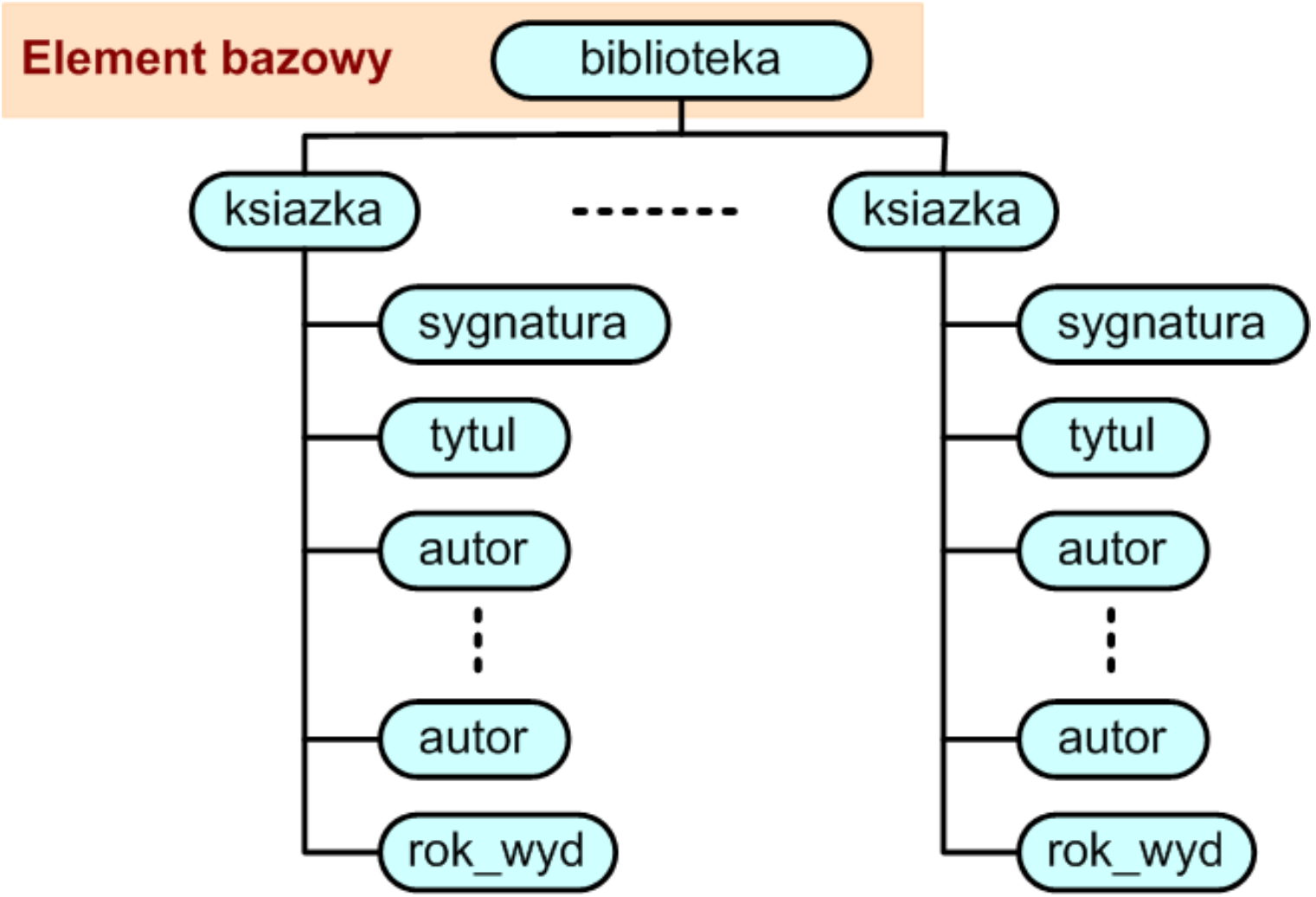
# Struktura drzewa (cd.)



# Element bazowy

- 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.

# Element bazowy (cd.)



# Element bazowy (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.)

- Wartości atrybutów muszą być umieszczone w cudzysłowie pojedynczym (') lub podwójnym (").
- Atrybuty są oddzielone spacjami i mogą mieć dowolną kolejność.
- Element może zwierać tylko jedno wystąpienie danego atrybutu.

# 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"/>
```

# Nazwy elementów i atrybutów

- Nazwa elementu lub atrybutu musi zaczynać się od litery albo znaku podkreślenia i może zawierać tylko znaki alfanumeryczne: litery, cyfry, myślniki (-), kropki (.) i znaki podkreślenia (_).
- Nazwy mogą zawierać także litery innych alfabetów.
- Niedozwolone są m.in. znaki spacji, równości, cudzysłowu, apostrofu, dolara, daszka, procentu, średnika.

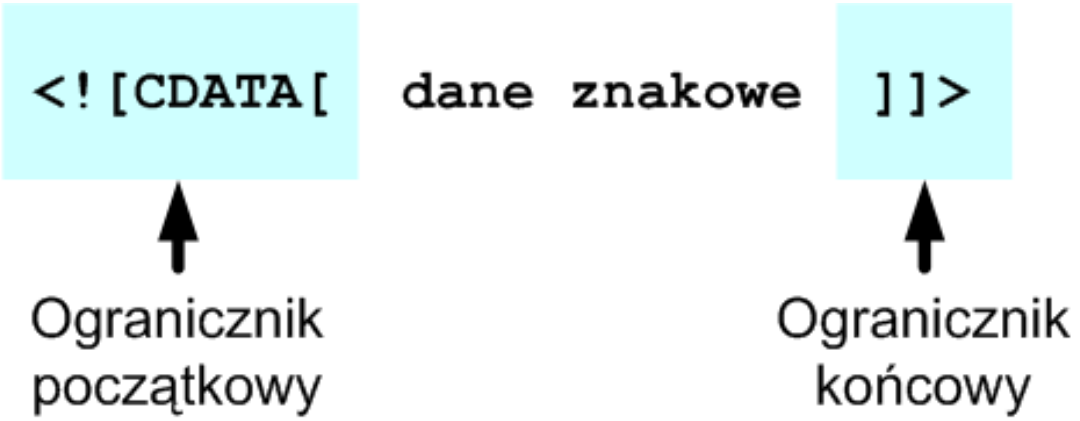
# Nazwy elementów i atrybutów (cd.)

- Użycie dwukropka jest dozwolone tylko dla przestrzeni nazw.
- Nazwa elementu w znaczniku początkowym musi być dokładnie taka sama jak w znaczniku końcowym.
- W nazwach XML wielkość liter ma znaczenie.

# 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.)



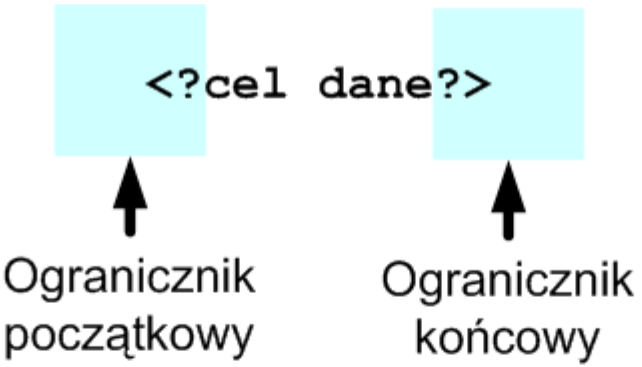

```
<rownanie numer=”20">
    <![CDATA[
    Jeśli a>=b wtedy c=10
    ]]>
</rownanie>
```

Sekcja CDATA

# Instrukcje przetwarzania

- Instrukcje przetwarzania służą do przekazywania informacji aplikacjom przetwarzającym dokumenty XML.
- Instrukcja przetwarzania zaczyna się od znaków <? i kończy się znakami ?>. Po znakach otwierających występuje nazwa obiektu docelowego (np. nazwa aplikacji) oraz dane dla aplikacji.
- Instrukcje przetwarzania nie mogą występować w obrębie znacznika.

# Instrukcje przetwarzania (cd.)




```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="plik.xsl"?>
```

Instrukcja przetwarzania

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

Cel

Dane

# 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.

# JSON

- JSON (*JavaScript Object Notation*):
  - samoopisujący się standard wymiany danych w aplikacjach internetowych,
  - sposób przekształcania obiektów JavaScript w łańcuch tekstowy i odwrotnie.
- Format JSON jest formatem tekstowym niezależnym od języka programowania.
- JSON oparty jest jest o:
  - zbiory par "atrybut-wartość",
  - uporządkowane listy wartości.

# Składnia JSON

- Dane są parami "atrybut-wartość".
- Dane są odseparowane od siebie za pomocą przecinka.
- Nawiasy klamrowe { } obejmują obiekt.
- Nawiasy prostokątne [ ] obejmują tablice.

# Składnia JSON

- Wartość JSON może być:
  - liczbą (całkowitą lub zmiennoprzecinkową),
  - łańcuchem znaków,
  - wartością logiczną (*true* albo *false*),
  - tablicą,
  - obiektem,
  - wartością pustą (*null*).

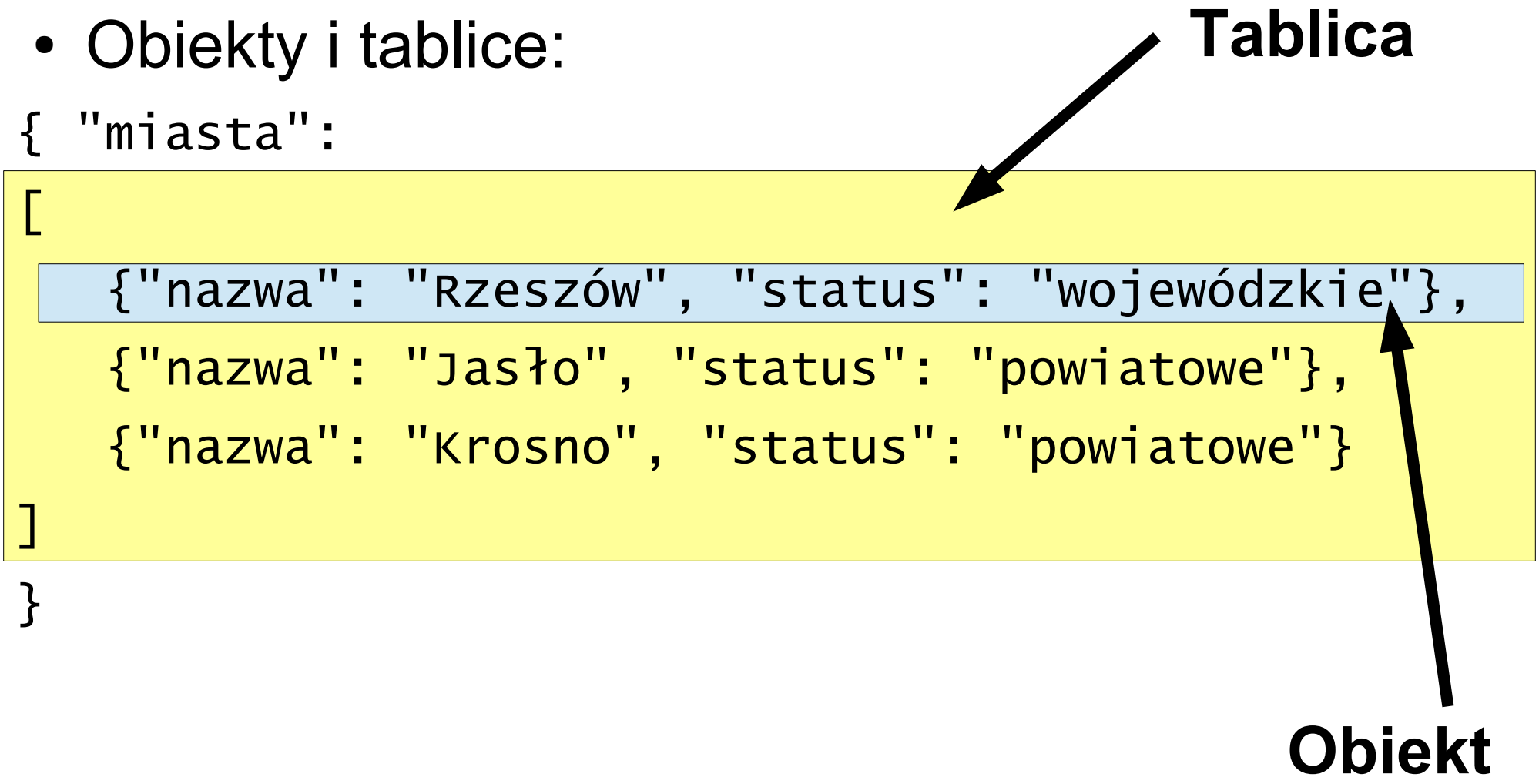
# Dane w formacie JSON

- Przykład danych w formacie JSON:

```
{ "miasta":[
    {"nazwa": "Rzeszów", "status": "wojewódzkie"},
    {"nazwa": "Jasło", "status": "powiatowe"},
    {"nazwa": "Krosno", "status": "powiatowe"}
]}
```

# Dane w formacie JSON

- Obiekty i tablice:



# Pliki JSON

- Dla plików JSON używane jest rozszerzenie "`.json`".
- Content-Type dla JSON to "`application/json`".

# JSON i XML

- Przykład danych w formacie XML:

```
<miasta>
   <miasto><nazwa>Rzeszów</nazwa><status>wojewódzkie</status></miasto>
   <miasto><nazwa>Jasło</nazwa><status>powiatowe</status></miasto>
   <miasto><nazwa>Krosno</nazwa><status>powiatowe</status></miasto>
</miasta>
```

- Przykład danych w formacie JSON:

```
{ "miasta": [
   {"nazwa": "Rzeszów", "status": "wojewódzkie"},
   {"nazwa": "Jasło", "status": "powiatowe"},
   {"nazwa": "Krosno", "status": "powiatowe"}
]}
```

# JSON i XML

- Zalety JSON w stosunku do XML:
  - JSON jest bardziej efektywny (używa mniejszej liczby znaków),
  - transformacje danych w formacie JSON na postać tekstową i odwrotnie jest szybsze niż przetwarzanie plików XML,
  - składnia JSON jest zgodna ze składnią języka JavaScript,
  - w JSON mogą być używane tablice.

# JSON i XML

- Używanie XML jest wskazane w przypadkach złożonych obiektów oraz w sytuacjach gdy istnieją zdefiniowane transformacje XSLT.

# JSON i XML

- Wspólne cechy formatów JSON i XML:
  - są samoopisujące się,
  - są hierarchiczne,
  - mogą być parowane w programach napisanych w różnych językach,
  - mogą być przesyłane za pomocą XMLHttpRequest.