

# Narzędzia informatyczne w językoznawstwie

## Perl - Przetwarzanie XML

Marcin Junczys-Dowmunt

`junczys@amu.edu.pl`

Zakład Logiki Stosowanej

<http://www.logic.amu.edu.pl>

7. maja 2008

- ▶ 3515 modułów w CPAN zawiera w nazwie skrót XML

- ▶ 3515 modułów w CPAN zawiera w nazwie skrót XML
- ▶ Perl jest językiem przeznaczonym do przetwarzania tekstu

- ▶ 3515 modułów w CPAN zawiera w nazwie skrót XML
- ▶ Perl jest językiem przeznaczonym do przetwarzania tekstu
- ▶ Dokumenty XML to pliki tekstowe

- ▶ 3515 modułów w CPAN zawiera w nazwie skrót XML
- ▶ Perl jest językiem przeznaczonym do przetwarzania tekstu
- ▶ Dokumenty XML to pliki tekstowe
- ▶ Perl jest silnie związany z WWW, podobnie jak XML

- ▶ Najbardziej podstawowy sposób tworzenia plików XML to generowanie dokumentów "na piechotę".

- ▶ Najbardziej podstawowy sposób tworzenia plików XML to generowanie dokumentów "na piechotę".
- ▶ Dokumenty XML są plikami tekstowymi, możemy więc korzystać ze standardowych funkcji Perla z zakresu przetwarzania tekstu.

- ▶ Najbardziej podstawowy sposób tworzenia plików XML to generowanie dokumentów "na piechotę".
- ▶ Dokumenty XML są plikami tekstowymi, możemy więc korzystać ze standardowych funkcji Perla z zakresu przetwarzania tekstu.
- ▶ W przypadku większych fragmentów tekstu, korzystanie z zapisu łańcuchów znakowych w cudzysłowie jest niewygodne.



- ▶ Najbardziej podstawowy sposób tworzenia plików XML to generowanie dokumentów "na piechotę".
- ▶ Dokumenty XML są plikami tekstowymi, możemy więc korzystać ze standardowych funkcji Perla z zakresu przetwarzania tekstu.
- ▶ W przypadku większych fragmentów tekstu, korzystanie z zapisu łańcuchów znakowych w cudzysłowie jest niewygodne.
- ▶ Można wtedy korzystać z tzw. *here-documents*.

- ▶ Najbardziej podstawowy sposób tworzenia plików XML to generowanie dokumentów "na piechotę".
- ▶ Dokumenty XML są plikami tekstowymi, możemy więc korzystać ze standardowych funkcji Perla z zakresu przetwarzania tekstu.
- ▶ W przypadku większych fragmentów tekstu, korzystanie z zapisu łańcuchów znakowych w cudzysłowie jest niewygodne.
- ▶ Można wtedy korzystać z tzw. *here-documents*.
- ▶ *here-document* to zapis, który mówi, że od nowej linii po identyfikatorze poprzedzonym znakami << aż do kolejnego napotkania identyfikatora należy traktować wszystko jako tekst ujęty w podwójnym cudzysłowie.

- ▶ Najbardziej podstawowy sposób tworzenia plików XML to generowanie dokumentów "na piechotę".
- ▶ Dokumenty XML są plikami tekstowymi, możemy więc korzystać ze standardowych funkcji Perla z zakresu przetwarzania tekstu.
- ▶ W przypadku większych fragmentów tekstu, korzystanie z zapisu łańcuchów znakowych w cudzysłowie jest niewygodne.
- ▶ Można wtedy korzystać z tzw. *here-documents*.
- ▶ *here-document* to zapis, który mówi, że od nowej linii po identyfikatorze poprzedzonym znakami << aż do kolejnego napotkania identyfikatora należy traktować wszystko jako tekst ujęty w podwójnym cudzysłowie.
- ▶ Wewnątrz *here-document* działa interpolacja zmiennych.

# Generowanie danych XML - Przykład

```
1 my @persons = (  
    { f_name => "Tomasz", s_name => "Kowalski" },  
    { f_name => "Bartosz", s_name => "Nowak" }  
);  
  
5  
my $xml = "<?xml version=\"1.0\"?>\n";  
$xml .= "<workers>\n";  
foreach my $person (@persons) {  
$xml .= <<XMLEND;  
10     <person>  
        <f_name>$person->{f_name}</f_name>  
        <s_name>$person->{s_name}</s_name>  
    </person>  
XMLEND  
15 }  
$xml .= "</workers>\n";  
  
print $xml;
```

# Wczytywanie danych XML - Co to jest parser?

- ▶ Przy generowanie dokumentu XML struktura danych narzuca strukturę dokumentu XML.

# Wczytywanie danych XML - Co to jest parser?

- ▶ Przy generowanie dokumentu XML struktura danych narzuca strukturę dokumentu XML.
- ▶ Odwrotnie struktura wczytywanego pliku XML ma wpływ na strukturę danych, która powstaje w pamięci komputera.

# Wczytywanie danych XML - Co to jest parser?

- ▶ Przy generowanie dokumentu XML struktura danych narzuca strukturę dokumentu XML.
- ▶ Odwrotnie struktura wczytywanego pliku XML ma wpływ na strukturę danych, która powstaje w pamięci komputera.
- ▶ Proces odtwarzania struktury hierarchicznej na podstawie danych linearnych nazywamy parsingiem.

# Wczytywanie danych XML - Co to jest parser?

- ▶ Przy generowanie dokumentu XML struktura danych narzuca strukturę dokumentu XML.
- ▶ Odwrotnie struktura wczytywanego pliku XML ma wpływ na strukturę danych, która powstaje w pamięci komputera.
- ▶ Proces odtwarzania struktury hierarchicznej na podstawie danych linearnych nazywamy parsingiem.
- ▶ W przypadku XML dotyczy to np. identyfikacji znaczników, zbudowania struktury zagnieżdżenia itp. na podstawie ciągu znaków.



# Wczytywanie danych XML - Co to jest parser?

- ▶ Przy generowanie dokumentu XML struktura danych narzuca strukturę dokumentu XML.
- ▶ Odwrotnie struktura wczytywanego pliku XML ma wpływ na strukturę danych, która powstaje w pamięci komputera.
- ▶ Proces odtwarzania struktury hierarchicznej na podstawie danych linearnych nazywamy parsingiem.
- ▶ W przypadku XML dotyczy to np. identyfikacji znaczników, zbudowania struktury zagnieżdżenia itp. na podstawie ciągu znaków.
- ▶ Parsing zajmuje się składniowym aspektem przetwarzania dokumentów XML.

# Wczytywanie danych XML - Co to jest parser?

- ▶ Przy generowanie dokumentu XML struktura danych narzuca strukturę dokumentu XML.
- ▶ Odwrotnie struktura wczytywanego pliku XML ma wpływ na strukturę danych, która powstaje w pamięci komputera.
- ▶ Proces odtwarzania struktury hierarchicznej na podstawie danych linearnych nazywamy parsingiem.
- ▶ W przypadku XML dotyczy to np. identyfikacji znaczników, zbudowania struktury zagnieżdżenia itp. na podstawie ciągu znaków.
- ▶ Parsing zajmuje się składniowym aspektem przetwarzania dokumentów XML.
- ▶ Rozróżnia się dwa paradygmaty parsingu XML: parsing odtwarzający strukturę drzewa XML w pamięci, parsing strumieniowy oparty na wydarzeniach.

- ▶ Cały dokument XML jest konwertowany na pojedynczą strukturę danych.

- ▶ Cały dokument XML jest konwertowany na pojedynczą strukturę danych.
- ▶ Podobnie jak sam dokument XML ta struktura danych ma strukturę drzewa, w którym węzły reprezentują elementy, atrybuty, dane tekstowe i inne fragmenty dokumentu XML.

- ▶ Cały dokument XML jest konwertowany na pojedynczą strukturę danych.
- ▶ Podobnie jak sam dokument XML ta struktura danych ma strukturę drzewa, w którym węzły reprezentują elementy, atrybuty, dane tekstowe i inne fragmenty dokumentu XML.
- ▶ Parser udostępnia funkcje służące do uzyskania dostępu do danych lub ich manipulacji, np. tworzenie i podczepianie nowych węzłów.

- ▶ Cały dokument XML jest konwertowany na pojedynczą strukturę danych.
- ▶ Podobnie jak sam dokument XML ta struktura danych ma strukturę drzewa, w którym węzły reprezentują elementy, atrybuty, dane tekstowe i inne fragmenty dokumentu XML.
- ▶ Parser udostępnia funkcje służące do uzyskania dostępu do danych lub ich manipulacji, np. tworzenie i podczepianie nowych węzłów.
- ▶ Cała struktura danych jest przechowywana w pamięci, w przypadku dużych dokumentów XML może to stanowić poważny problem. Struktura danych może zajmować do 30 razy tyle pamięci co postać tekstowa dokumentu XML.

- ▶ Cały dokument XML jest konwertowany na pojedynczą strukturę danych.
- ▶ Podobnie jak sam dokument XML ta struktura danych ma strukturę drzewa, w którym węzły reprezentują elementy, atrybuty, dane tekstowe i inne fragmenty dokumentu XML.
- ▶ Parser udostępnia funkcje służące do uzyskania dostępu do danych lub ich manipulacji, np. tworzenie i podczepianie nowych węzłów.
- ▶ Cała struktura danych jest przechowywana w pamięci, w przypadku dużych dokumentów XML może to stanowić poważny problem. Struktura danych może zajmować do 30 razy tyle pamięci co postać tekstowa dokumentu XML.
- ▶ Takie parseery sprawdzają poprawność dokumentu XML zanim udostępnią dane.

- ▶ Parseery strumieniowe nie budują drzewa w pamięci, tylko generują strumień zdarzeń, które są przechwytywane oraz przetwarzane na bieżąco przez tzw. funkcje *callback*.



- ▶ Parseery strumieniowe nie budują drzewa w pamięci, tylko generują strumień zdarzeń, które są przechwytywane oraz przetwarzane na bieżąco przez tzw. funkcje *callback*.
- ▶ Różnym zdarzeniom przydziela się różne funkcje *callback*.

- ▶ Parseery strumieniowe nie budują drzewa w pamięci, tylko generują strumień zdarzeń, które są przechwytywane oraz przetwarzane na bieżąco przez tzw. funkcje *callback*.
- ▶ Różnym zdarzeniom przydziela się różne funkcje *callback*.
- ▶ Odpowiednie zdarzenia są generowane w momentach gdy parser w trakcie wczytywania natrafia np. na znaczniki otwierające, znaczniki zamykające itp.

- ▶ Parseery strumieniowe nie budują drzewa w pamięci, tylko generują strumień zdarzeń, które są przechwytywane oraz przetwarzane na bieżąco przez tzw. funkcje *callback*.
- ▶ Różnym zdarzeniom przydziela się różne funkcje *callback*.
- ▶ Odpowiednie zdarzenia są generowane w momentach gdy parser w trakcie wczytywania natrafia np. na znaczniki otwierające, znaczniki zamykające itp.
- ▶ Parser udostępnia dane z dokumentu XML w trakcie przetwarzania dokumentu. Nie musi wczytać całość dokumentu.

- ▶ Parseery strumieniowe nie budują drzewa w pamięci, tylko generują strumień zdarzeń, które są przechwytywane oraz przetwarzane na bieżąco przez tzw. funkcje *callback*.
- ▶ Różnym zdarzeniom przydziela się różne funkcje *callback*.
- ▶ Odpowiednie zdarzenia są generowane w momentach gdy parser w trakcie wczytywania natrafia np. na znaczniki otwierające, znaczniki zamykające itp.
- ▶ Parser udostępnia dane z dokumentu XML w trakcie przetwarzania dokumentu. Nie musi wczytać całość dokumentu.
- ▶ Parser strumieniowy udostępnia dane zanim sprawdzi poprawność składniową dokumentu.

- ▶ Parseery strumieniowe nie budują drzewa w pamięci, tylko generują strumień zdarzeń, które są przechwytywane oraz przetwarzane na bieżąco przez tzw. funkcje *callback*.
- ▶ Różnym zdarzeniom przydziela się różne funkcje *callback*.
- ▶ Odpowiednie zdarzenia są generowane w momentach gdy parser w trakcie wczytywania natrafia np. na znaczniki otwierające, znaczniki zamykające itp.
- ▶ Parser udostępnia dane z dokumentu XML w trakcie przetwarzania dokumentu. Nie musi wczytać całość dokumentu.
- ▶ Parser strumieniowy udostępnia dane zanim sprawdzi poprawność składniową dokumentu.
- ▶ Wymagania pamięciowe takich parserów są niskie i możemy przetwarzać wielkie dokumenty XML, niestety obsługa jest bardziej skomplikowana.

- ▶ Standardowymi interfejsami do parserów XML dla różnych języków

- ▶ Standardowymi interfejsami do parserów XML dla różnych języków programowania są np. DOM (Document Object Model) oraz SAX (Simple API for XML).
- ▶ Odpowiadają im moduły XML::DOM oraz XML::SAX.

- ▶ Standardowymi interfejsami do parserów XML dla różnych języków programowania są np. DOM (Document Object Model) oraz SAX (Simple API for XML).
- ▶ Odpowiadają im moduły XML::DOM oraz XML::SAX.
- ▶ XML::DOM to interfejs do drzewiastej struktury dokumentu.



- ▶ Standardowymi interfejsami do parserów XML dla różnych języków programowania są np. DOM (Document Object Model) oraz SAX (Simple API for XML).
- ▶ Odpowiadają im moduły XML::DOM oraz XML::SAX.
- ▶ XML::DOM to interfejs do drzewiastej struktury dokumentu.
- ▶ XML::SAX obsługuje parsery strumieniowe.

# Przykład dokumentu XML (leksykon.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<leksykon>
  <wpis nr="1">
    <lemat>analiza</lemat>
    <informacje>
      <część_mowy>rzeczownik</część_mowy>
      <rodzaj>żeński</rodzaj>
      <semantyka>process</semantyka>
    </informacje>
    <formy>
      <forma p="mian" l="1">analiza</forma>
      <forma p="dop" l="1">analizy</forma>
      <forma p="cel" l="1">analizie</forma>
      ...
    </formy>
  </wpis>
  ...
</leksykon>
```

# Moduł XML::DOM

```
1 use XML::DOM;
  use Data::Dumper;
  my $parser = new XML::DOM::Parser;
  my $doc = $parser->parsefile("leksykon.xml");

5
  my $nodes = $doc->getElementsByTagName ("forma");
  my $n = $nodes->getLength();

  for(my $i = 0; $i < $n; $i++) {
10     my $text = $nodes->item($i)->getFirstChild();
     my $forma = $text->getNodeValue();
     my $przy = $nodes->item($i)->getAttribute("p");
     print "Forma: $forma - Przypadek: $przy\n";
  }
```

# Moduł XML::SAX

```
1 use XML::SAX::Expat;
my $parser = XML::SAX::Expat->new(Handler => MyHandler->new());
my $ref = $parser->parse_file("leksykon.xml");

5 package MyHandler;
sub new {
    my $type = shift;
    my $obj = { data => {} };
    return bless $obj, $type;
10 }

sub start_element {
my( $self, $prop ) = @_;
if($prop->{Name} eq "forma") {
15     $self->{data}->{last} = "forma";
    $self->{data}->{text} = "";
    $self->{data}->{przypadek} = $prop->{'Attributes'}->{"{}p"}->{'Value'};
} }

20 sub characters {
my( $self, $prop ) = @_;
if ( $self->{data}->{last} eq "forma" ) {
    $self->{data}->{text} .= $prop->{Data};
} }

25 sub end_element {
my( $self, $prop ) = @_;
if($prop->{Name} eq "forma") {
    print "Forma: $self->{data}->{text} - Przypadek: $self->{data}->{przypadek}\n";
30     $self->{data}->{last} = "";
} } }
```

- ▶ XML::Simple został stworzony z myślą o plikach konfiguracyjnych zapisanych w XML.

- ▶ XML::Simple został stworzony z myślą o plikach konfiguracyjnych zapisanych w XML.
- ▶ Moduł ten może służyć do prostych zadań związanych z przetwarzaniem XML.

- ▶ XML::Simple został stworzony z myślą o plikach konfiguracyjnych zapisanych w XML.
- ▶ Moduł ten może służyć do prostych zadań związanych z przetwarzaniem XML.
- ▶ Jest to interfejs oparty na strukturze drzewiastej dokumentu.

- ▶ XML::Simple został stworzony z myślą o plikach konfiguracyjnych zapisanych w XML.
- ▶ Moduł ten może służyć do prostych zadań związanych z przetwarzaniem XML.
- ▶ Jest to interfejs oparty na strukturze drzewiastej dokumentu.
- ▶ Nie ma jednak typowego interfejsu jak w przypadku XML::DOM.



- ▶ XML::Simple został stworzony z myślą o plikach konfiguracyjnych zapisanych w XML.
- ▶ Moduł ten może służyć do prostych zadań związanych z przetwarzaniem XML.
- ▶ Jest to interfejs oparty na strukturze drzewiastej dokumentu.
- ▶ Nie ma jednak typowego interfejsu jak w przypadku XML::DOM.
- ▶ Dostęp do węzłów odbywa się za pomocą zwykłych zagnieżdżonych struktur danych, czyli anonimowych tablic i haszów.

- ▶ XML::Simple został stworzony z myślą o plikach konfiguracyjnych zapisanych w XML.
- ▶ Moduł ten może służyć do prostych zadań związanych z przetwarzaniem XML.
- ▶ Jest to interfejs oparty na strukturze drzewiastej dokumentu.
- ▶ Nie ma jednak typowego interfejsu jak w przypadku XML::DOM.
- ▶ Dostęp do węzłów odbywa się za pomocą zwykłych zagnieżdżonych struktur danych, czyli anonimowych tablic i haszów.
- ▶ XML::Simple nie radzi sobie z mieszaną treścią, tzn. gdy element zawiera tekst w którym pojawiają się inne elementy. Np. `<element> To jest <b>przykład</b> </element>` zwraca dziwne wyniki.

```
1 use XML::Simple;

my $xml = new XML::Simple(ForceArray => 1);
my $data = $xml->XMLin("leksykon.xml");

5 foreach my $wpis (@{$data->{wpis}}) {
    my $formy = $wpis->{formy}->[0]->{forma};
    foreach my $forma (@$formy) {
        my $f = $forma->{content};
10        my $p = $forma->{p};
        print "Forma: $f - Przypadek: $p\n";
    }
}
```

# Wczytywanie dokumentów XML za pomocą XML::Simple

```
1  $VAR1 = {
      'wpis' => [ {
          'informacje' => [ {
              'semantyka' => [ 'process' ],
              'czesc_mowy' => [ 'rzeczownik' ],
              'rodzaj' => [ 'meski' ]
          } ],
          'formy' => [ {
              'forma' => [ {
                  'l' => '1',
                  'p' => 'mian',
                  'content' => 'analiza'
                },
                {
                  'l' => '1',
                  'p' => 'dop',
                  'content' => 'analizy'
                },
                {
                  'l' => '1',
                  'p' => 'cel',
                  'content' => 'analizie'
                }
              ]
            },
          'lemat' => [ 'analiza' ],
          'nr' => '1'
        } ]
    };
```

# Tworzenie dokumentów XML za pomocą XML::Simple

```
1 use XML::Simple;
   use Data::Dumper;

   my $xml = new XML::Simple(ForceArray => 1);
5 my $data = $xml->XMLin("leksykon.xml");

   print Dumper($data);

   print $xml->XMLout($data);
```

# Kontrolowanie działania modułu XML::Simple

```
1 use XML::Simple;

my $xml = new XML::Simple(
    ForceArray => [ "wpis" ],
5     GroupTags => { "formy" => "forma" },
);

my $data = $xml->XMLin("lex.xml");

10 foreach my $wpis (@{$data->{wpis}}) {
    foreach my $forma (@{$wpis->{formy}}) {
        my $f = $forma->{content};
        my $p = $forma->{p};
        print "Forma: $f - Przypadek: $p\n";
15     }
}
```

# Wczytywanie dokumentów XML za pomocą XML::Simple

```
1  $VAR1 = {  
    'wpis' => [ {  
        'informacje' => {  
            'semantyka' => 'process',  
            'czesc.mowy' => 'rzeczownik',  
            'rodzaj' => 'meski'  
        },  
        'formy' => [ {  
            'l' => '1',  
            'p' => 'mian',  
            'content' => 'analiza'  
        },  
        {  
            'l' => '1',  
            'p' => 'dop',  
            'content' => 'analizy'  
        },  
        {  
            'l' => '1',  
            'p' => 'cel',  
            'content' => 'analizie'  
        } ],  
        'lemat' => 'analiza',  
        'nr' => '1'  
    } ]  
};
```

## XML::Twig

- ▶ XML::Twig jest jakby hybrydą parsera opartego na drzewach oraz parsera opartego na strumieniach zdarzeń.



## XML::Twig

- ▶ XML::Twig jest jakby hybrydą parsera opartego na drzewach oraz parsera opartego na strumieniach zdarzeń.
- ▶ Pozwala na tworzenie struktur drzewiastych, ale może być ograniczony do określonych podelementów.

## XML::Twig

- ▶ XML::Twig jest jakby hybrydą parsera opartego na drzewach oraz parsera opartego na strumieniach zdarzeń.
- ▶ Pozwala na tworzenie struktur drzewiastych, ale może być ograniczony do określonych podelementów.
- ▶ W ten sposób umożliwia wczytywanie dużych dokumentów, dla których są tworzona gałęzie (twigs) a nie całe drzewa.

## XML::Twig

- ▶ XML::Twig jest jakby hybrydą parsera opartego na drzewach oraz parsera opartego na strumieniach zdarzeń.
- ▶ Pozwala na tworzenie struktur drzewiastych, ale może być ograniczony do określonych podelementów.
- ▶ W ten sposób umożliwia wczytywanie dużych dokumentów, dla których są tworzone gałęzie (twigs) a nie całe drzewa.

## XML::Dumper

- ▶ Moduł podobny do Data::Dumper oraz XML::Simple

## XML::Twig

- ▶ XML::Twig jest jakby hybrydą parsera opartego na drzewach oraz parsera opartego na strumieniach zdarzeń.
- ▶ Pozwala na tworzenie struktur drzewiastych, ale może być ograniczony do określonych podelementów.
- ▶ W ten sposób umożliwia wczytywanie dużych dokumentów, dla których są tworzone gałęzie (twigs) a nie całe drzewa.

## XML::Dumper

- ▶ Moduł podobny do Data::Dumper oraz XML::Simple
- ▶ Tworzy XML-ową reprezentację struktur danych w Perlu.

## XML::Twig

- ▶ XML::Twig jest jakby hybrydą parsera opartego na drzewach oraz parsera opartego na strumieniach zdarzeń.
- ▶ Pozwala na tworzenie struktur drzewiastych, ale może być ograniczony do określonych podelementów.
- ▶ W ten sposób umożliwia wczytywanie dużych dokumentów, dla których są tworzona gałęzie (twigs) a nie całe drzewa.

## XML::Dumper

- ▶ Moduł podobny do Data::Dumper oraz XML::Simple
- ▶ Tworzy XML-ową reprezentację struktur danych w Perlu.
- ▶ Konstrukcje nie zawsze czytelne (np. struktury cykliczne)

## XML::Twig

- ▶ XML::Twig jest jakby hybrydą parsera opartego na drzewach oraz parsera opartego na strumieniach zdarzeń.
- ▶ Pozwala na tworzenie struktur drzewiastych, ale może być ograniczony do określonych podelementów.
- ▶ W ten sposób umożliwia wczytywanie dużych dokumentów, dla których są tworzona gałęzie (twigs) a nie całe drzewa.

## XML::Dumper

- ▶ Moduł podobny do Data::Dumper oraz XML::Simple
- ▶ Tworzy XML-ową reprezentację struktur danych w Perlu.
- ▶ Konstrukcje nie zawsze czytelne (np. struktury cykliczne)
- ▶ Potrafi odtworzyć oryginalną strukturę danych z pliku XML-owego.