SZKOLENIE XML I XSLT W INSPIRE (POZIOM ZAAWANSOWANY)

Agnieszka Chojka

Warszawa, listopad 2018

MINISTERSTWO ŚRODOWISKA

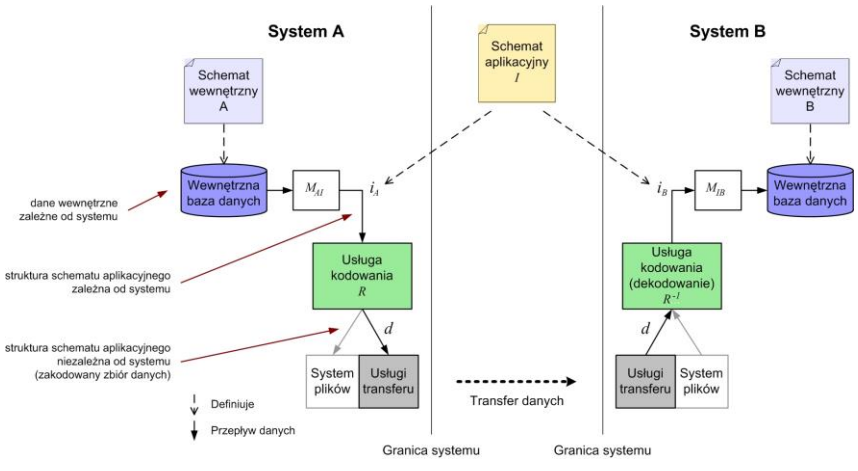
Sfinansowano ze środków Narodowego Funduszu Ochrony Środowiska i Gospodarki Wodnej

TRAPPA

ZAKRES TEMATYCZNY SZKOLENIA

- **Wprowadzenie**
- **Podstawy gramatyki języka XML (przypomnienie)**
- **Podstawy gramatyki języka XML Schema (przypomnienie)**
- **Narzędzia wspomagające tworzenie plików XML i XSD**
- **Podstawy gramatyki języków XLink, XPointer, XPath (przypomnienie)**
- **Podstawy gramatyki języka XSLT**
- **Język Schematron (bonus)**

INTEROPERACYJNA WYMIANA DANYCH



[źródło: ISO/TC 211, 2011. ISO 19118 Geographic information – Encoding]

ISO 19118

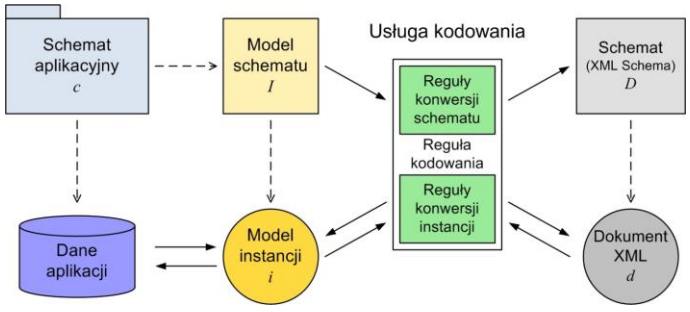
- reguły kodowania wykorzystywane podczas wymiany danych przestrzennych
  - **reguła kodowania** pozwala na zakodowanie informacji geograficznej
    - zdefiniowanej przez schematy aplikacyjne i schematy znormalizowane

    na strukturę danych niezależną od systemu
    - odpowiednią do przesyłania i przechowywania danych
  - na potrzeby neutralnej wymiany danych normy ISO serii 19100 zalecają reguły kodowania oparte na **języku XML**
    - niezależny od platformy informatycznej
    - wykazuje interoperacyjność z siecią www

WPROWADZENIE

REGUŁA KODOWANIA XML

- reguły konwersji **schematu aplikacyjnego UML** na **schemat struktur danych zapisany w XML Schema**
- reguły konwersji **instancji** na **struktury danych zapisane w dokumencie XML**



[źródło: ISO/TC 211, 2011. ISO 19118 Geographic information – Encoding]

REGUŁA KONWERSJI SCHEMATU

- zapewnia, że **dokumenty XML** wytworzone przy użyciu reguł konwersji danych (instancji) będą poprawne
- definiuje jak utworzyć **dokument schematu XML** zgodnie ze **schematem aplikacyjnym** wyrażonym w **UML**
  - **schemat XML** = plik **XSD** (ang. *XML Schema Definition*)
    - powinien zawierać definicje typów, deklaracje atrybutów i elementów, które odpowiadają klasom zdefiniowanym w schemacie aplikacyjnym
    - fizycznie może być pojedynczym dokumentem schematu lub może być podzielony na kilka oddzielnych dokumentów

PODSTAWY GRAMATYKI JĘZYKA XML

PRZYPOMNIENIE

JĘZYK XML

- ang. *e**X**tensible **M**arkup **L**anguage*
- rozszerzalny język znaczników
  - język znaczników podobnie jak język HTML (ang. *H**yper**T**ext **M**arkup **L**anguage*)
- standard opracowany przez W3C
- zaprojektowany do przesyłania i przechowywania danych
- niezależne programowo i sprzętowo „narzędzie” przenoszenia informacji
  - HTML odpowiada jedynie za wyświetlanie informacji
  - XML odpowiada za transfer informacji
- dokumenty XML zapisywane w plikach z rozszerzeniem *.XML*

JĘZYK XML

DOKUMENT XML | ZASADY SKŁADNI

- powinien zaczynać się od deklaracji XML
- musi mieć jeden unikalny element główny („korzeń”, ang. *root*)
  - „rodzic” dla pozostałych elementów „dzieci”
- znaczniki otwierające/początkowe (np. <notatka>) muszą posiadać odpowiadające im znaczniki zamykające/końcowe (np. </notatka>)
  - wyjątek: element pusty <notatka/>
- znaczniki rozróżniają małe i duże litery
- wszystkie elementy muszą być zamknięte
- wszystkie elementy muszą być odpowiednio zagnieżdżone
- wszystkie wartości atrybutów muszą być ujęte w cudzysłów

DOKUMENT XML | PRZYKŁAD

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<notatka>
  <dla>Tomka</dla>
  <od>Ani</od>
  <tytuł>Przypomnienie</tytuł>
  <treść>Nie zapomnij o mnie podczas weekendu!</treść>
</notatka>
```

- wiersz 1 (deklaracja XML)
  - definiuje wersję XML (1.0) i stosowane kodowanie (ISO-8859-2, zbiór znaków dla Europy Środkowej i Wschodniej)
- wiersz 2
  - opisuje element główny („korzeń”) dokumentu XML: <notatka>
- wiersze 3-6
  - opisują 4 elementy „dzieci” elementu głównego: <dla>, <od>, <tytuł>, <treść>
- wiersz 7
  - definiuje koniec elementu głównego: </notatka>

XML VS HTML | PRZYKŁAD

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<notatka>
  <dla>Tomka</dla>
  <od>Ani</od>
  <tytuł>Przypomnienie</tytuł>
  <treść>Nie zapomnij o mnie podczas weekendu!</treść>
</notatka>
```

dokument XML

```
<?xml version="1.0" encoding="ISO-8859-2" ?>
<notatka>
  <dla>Tomka</dla>
  <od>Ani</od>
  <tytuł>Przypomnienie</tytuł>
  <treść>Nie zapomnij o mnie podczas weekendu!</treść>
</notatka>
```

widok dokumentu XML w przeglądarce internetowej

```
<html>
  <head>
    <title>Notatka</title>
  </head>
  <body>
    <table align="center" width="350" bgcolor="red">
      <tr><td align="center"><h2><u>PRZYPOMNIENIE</u></h2></td></tr>
      <tr><td align="center"><h3>Nie zapomnij o mnie podczas weekendu!</h3>
      <tr><td align="right"><h3><i>Ania</i></h3></td></tr>
    </table>
  </body>
</html>
```

dokument HTML

widok dokumentu HTML w przeglądarce internetowej

DOKUMENT XML | ELEMENTY I ATRYBUTY

- **element XML**
  - wszystko to, co znajduje się między znacznikiem początkowym (łącznie z nim) elementu a znacznikiem końcowym (łącznie z nim) elementu
  - może zawierać
    - inne elementy
    - tekst
    - atrybuty
    - lub wszystkie powyższe
- **atrybut XML**
  - dostarcza dodatkowe informacje o elementach XML

ELEMENTY I ATRYBUTY XML | PRZYKŁAD

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<księgarnia>
  <książka kategoria="DZIECI">
    <tytuł>Harry Potter i czara ognia</tytuł>
    <autor>J K. Rowling</autor>
    <rok>2001</rok>
    <cena>49.99</cena>
  </książka>
  <książka kategoria="KRYMINAŁ">
    <tytuł>Byczki w pomidorach</tytuł>
    <autor>J. Chmielewska</autor>
    <rok>2010</rok>
    <cena>38.99</cena>
  </książka>
</księgarnia>
```

- element główny <księgarnia> zawiera element <książka>
- element <książka> składa się z elementów: <tytuł>, <autor>, <rok> i <cena>, które zawierają tekst
- element <książka> dodatkowo posiada atrybut „kategoria”

ELEMENTY XML | ZASADY SKŁADNI

- nazwy mogą zawierać litery, liczby i inne znaki
- nazwy nie mogą rozpoczynać się od liczby lub znaku przestankowego (np. . , : ; „ ‘ ? -)
- nazwy nie mogą rozpoczynać się od liter xml (lub XML, lub Xml, itp.)
- nazwy nie mogą zawierać spacji
- każda nazwa może być użyta, żadne słowa nie są zarezerwowane (zastrzeżone)
- zaleca się, aby nazwa elementu była opisowa, krótka i prosta
  - najlepiej stosować znak podkreślenia (np. <tytuł_książki>)
  - należy unikać znaków: „-”, „.” i „:”
    - mogą zostać błędnie zinterpretowane przez różne programy

ELEMENT VS ATRYBUT | PRZYKŁAD

- ta sama informacja, ale zapisana na różne sposoby

```
<osoba>
  <płeć>kobieta</płeć>
  <imię>Anna</imię>
  <nazwisko>Nowak</nazwisko>
</osoba>
```

element płeć

```
<osoba płeć="kobieta">
  <imię>Anna</imię>
  <nazwisko>Nowak</nazwisko>
</osoba>
```

atrybut płeć

- do zapisu
  - **danych** najlepiej stosować **elementy**
  - **dodatkowych informacji** o elementach najlepiej stosować **atrybuty**

PRZESTRZEŃ NAZW

- nazwy elementów w XML są definiowane przez użytkownika
  - może to prowadzić do konfliktów nazw elementów pochodzących z różnych dokumentów XML
- aby temu zapobiec stosuje się **przedrostek nazwy**, dla którego musi być zdefiniowana tzw. **przestrzeń nazw** (ang. *namespace*)
  - określana przez atrybut **xmlns** w znaczniku rozpoczynającym element
    - **xmlns:prefix="URI"**

URI

- ang. ***U**niform **R**esource **I**dentifier*
- unikalny identyfikator zasobu w sieci
  - URN (ang. ***U**niform **R**esource **N**ame*)
    - nazwa zasobu w sieci
    - np. *urn:x-inspire:specification:gmlas:BaseTypes:3.2*
  - URL (ang. ***U**niform **R**esource **L**ocator*)
    - adres zasobu w sieci
    - np. *http://www.opengis.net/gml/3.2*
  - **przestrzeń nazw URI** nie jest używana do szukania informacji
    - jej zadaniem jest nadawanie unikalnych nazw przestrzeni nazw

PRZESTRZEŃ NAZW | PRZYKŁAD

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<dom>
 <j:stół xmlns:j="http://www.sklep.meblowy.com/jadalnia">
  <j:nazwa>Olgierd</j:nazwa>
   <j:szerokość>80</j:szerokość>
   <j:długość>120</j:długość>
 </j:stół>
 <n:stół xmlns:n="http://www.warsztat.com/narzędzia">
  <n:garaż>
   <n:narzędzia>imadło</n:narzędzia>
   <n:narzędzia>młotek</n:narzędzia>
  </n:garaż>
 </n:stół>
</dom>
```

- w znaczniku <stół> atrybut xmlns otrzymał przedrostki „j:” i „n:” określające odpowiednie przestrzenie nazw
  - jeśli dla elementu zostanie zdefiniowana przestrzeń nazw, wszystkie elementy „dzieci” z tym samym przedrostkiem są powiązane z tą samą przestrzenią nazw

PODSTAWY GRAMATYKI JĘZYKA XML SCHEMA

PRZYPOMNIENIE

XML SCHEMA

- ang. *XML Schema*
- schemat XML, schemat rozszerzalnego języka znaczników

- standard opracowany przez W3C
- opisuje strukturę dokumentu XML
- wykorzystuje składnię języka XML
- dokumenty zawierające definicje XML Schema zapisywane w plikach z rozszerzeniem *.XSD* *(ang. **X**ML **S**chema **D**efinition)*

DOKUMENT XML SCHEMA | ZASADY SKŁADNI

- definiuje
  - elementy, które mogą pojawić się w dokumencie XML
  - atrybuty, które mogą pojawić się w dokumencie XML
  - które elementy są elementami „dziećmi”
  - kolejność (porządek) elementów „dzieci”
  - liczbę elementów „dzieci”
  - czy element jest pusty, czy może zawierać tekst
  - typy danych dla elementów i atrybutów XML
  - wartości domyślne i stałe dla elementów i atrybutów XML

DOKUMENT XML SCHEMA | ZASADY SKŁADNI

- elementem głównym („korzeniem”) każdego dokumentu XML Schema jest element **<schema>**
  - może on zawierać
    - atrybuty
    - **elementy globalne**
      - elementy będące bezpośrednio „dziećmi” elementu <schema>
    - **elementy lokalne**
      - elementy zagnieżdżone wewnątrz innych elementów

DOKUMENT XML SCHEMA | PRZYKŁAD

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">
...
</xs:schema>
```

- *xmlns:xs="http://www.w3.org/2001/XMLSchema"*
  - elementy i typy danych użyte w schemacie pochodzą z przestrzeni nazw *http://www.w3.org/2001/XMLSchema*
  - elementy i typy danych, które pochodzą z tej przestrzeni nazw powinny być poprzedzone przedrostkiem *xs*
- targetNamespace="http://www.w3schools.com"
  - elementy zdefiniowane przez schemat (np. <książka>, <tytuł>, <autor>) pochodzą z przestrzeni nazw *http://www.w3schools.com*
- xmlns="http://www.w3schools.com"
  - domyślną przestrzenią nazw jest *http://www.w3schools.com*
- elementFormDefault="qualified"
  - każdy element użyty w instancji (egzemplarzu) dokumentu XML, który został zadeklarowany w schemacie musi mieć określoną przestrzeń nazw

DOKUMENT XML | PRZYKŁAD

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<księgarnia xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com księgarnia.xsd">
...
</księgarnia>
```

- *xmlns="http://www.w3schools.com"*
  - określa domyślną deklarację przestrzeni nazw, która oznacza, że wszystkie elementy użyte w tym dokumencie XML są zadeklarowane w przestrzeni nazw http://www.w3schools.com
- jeżeli przestrzeń nazw instancji XML Schema jest dostępna *xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"*, można zastosować atrybut *schemaLocation*, który posiada dwie wartości
  - przestrzeń nazw
  - lokalizacja schematu XML dla tej przestrzeni nazw *xsi:schemaLocation="http://www.w3schools.com ksiegarnia.xsd"*

JĘZYK XML SCHEMA

DOKUMENT XML SCHEMA & DOKUMENT XML | PRZYKŁAD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
attributeFormDefault="unqualified">

  <xs:element name="Ewidencja_Jaskin">
    …
  </xs:element>

</xs:schema>
```

dokument XML Schema (plik XSD)

```
<?xml version="1.0" encoding="UTF-8"?>
<Ewidencja_Jaskin
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Jaskinie.xsd">

…

</Ewidencja_Jaskin>
```

dokument XML (plik XML)

DOKUMENT XML SCHEMA | ZAWARTOŚĆ

- **element prosty** (typ prosty)
  - simpleType
- **element złożony** (typ złożony)
  - complexType

ELEMENT SIMPLETYPE (TYP PROSTY)

- **element prosty**
  - element XML, który
    - zawiera tylko tekst
    - nie może zawierać żadnych innych elementów i atrybutów

```
<gco:CharacterString>metadane@pgi.gov.pl</gco:CharacterString>
```

  - składania definicji elementu prostego **<xs:element name="xxx" type="yyy"/>**
    - "xxx" nazwa elementu
    - "yyy" typ danych tego elementu

```
<xs:element  name="electronicMailAddress"
             type="gco:CharacterString_PropertyType"
             minOccurs="0" maxOccurs="unbounded"/>
```

ELEMENT SIMPLETYPE (TYP PROSTY)

- XML Schema posiada wiele wbudowanych typów danych, np.
  - xs:string
  - xs:decimal
  - xs:integer
  - xs:boolean
  - xs:date
  - xs:time

```
<xs:element name="CharacterString" type="xs:string"/>
<xs:element name="Real" type="xs:double"/>
<xs:element name="DateTime" type="xs:dateTime"/>
```

- element prosty może mieć również określoną wartość
  - domyślną (**default**)
  - stałą (**fixed**)

JĘZYK XML SCHEMA

ELEMENT SIMPLETYPE (TYP PROSTY)

- w XML Schema wszystkie **atrybuty** są zadeklarowane jako typy proste
- elementy proste nie mogą mieć atrybutów
  - element, który posiada atrybuty jest typu złożonego (**complexType**)
- atrybut (sam w sobie) jest zawsze zadeklarowany jako typ prosty (**simpleType**)

ELEMENT SIMPLETYPE (TYP PROSTY)

- **atrybut**
  - składania definiowania atrybutu
    <xs:attribute name="xxx" type="yyy"/>
    - "xxx" nazwa atrybutu
    - "yyy" typ danych atrybutu

```
<xs:attribute name="isInfinite" type="xs:boolean"/>
```

  - może
    - mieć określoną wartość domyślną lub stałą
    - być opcjonalny lub wymagany (**use="required"**)
      - domyślnie atrybut jest opcjonalny

JĘZYK XML SCHEMA

ELEMENT SIMPLETYPE (TYP PROSTY)

- w XML Schema można zdefiniować ograniczenie (**restriction**) zawartości elementu lub atrybutu
  - stosowane do określania akceptowalnych wartości elementów i atrybutów

Ograniczenie	Opis
enumeration	definiuje listę dopuszczalnych wartości
fractionDigits	określa max liczbę dozwolonych miejsc dziesiętnych (musi być większe lub równe 0)
length	określa dokładną liczbę znaków lub listę elementów dozwolonych (musi być większe lub równe 0)
maxExclusive	określa górną granicę dla wartości numerycznych (jej wartość musi być mniejsza niż ta wartość)
maxInclusive	określa górną granicę dla wartości numerycznych (jej wartość musi być mniejsza lub równa tej wartości)
maxLength	określa max liczbę znaków lub listę elementów dozwolonych (musi być większe lub równe 0)
minExclusive	określa dolną granicę dla wartości numerycznych (jej wartość musi być większa niż ta wartość)
minInclusive	określa dolną granicę dla wartości numerycznych (jej wartość musi być większa lub równa tej wartości)
minLength	określa min liczbę znaków lub listę elementów dozwolonych (musi być większe lub równe 0)
pattern	definiuje dokładną sekwencję dopuszczalnych znaków
totalDigits	określa dokładną liczbę dozwolonych cyfr (musi być większe od 0)
whiteSpace	określa jak „białe" znaki są obsługiwane (whitespace: przesunięcia o wiersz, tabulatory, spacje i powroty karetki)

ELEMENT SIMPLETYPE (TYP PROSTY) | PRZYKŁAD

```
<xs:element name="hasło">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="5"/>
      <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- element prosty "hasło" z ograniczeniem zawartości
  - długość "hasła" musi wynosić
    co najmniej 5 i co najwyżej 8 znaków

JĘZYK XML SCHEMA

ELEMENT COMPLEXTYPE (TYP ZŁOŻONY)

- **element złożony**
  - element XML, który zawiera
    - inne elementy
    - i/lub atrybuty
  - wyróżnia się 4 rodzaje elementów złożonych
    (każdy z nich może również zawierać atrybuty)
    - elementy puste
    - elementy, które zawierają tylko inne elementy
    - elementy, które zawierają tylko tekst
    - elementy, które zawierają zarówno inne elementy jak i tekst

ELEMENT COMPLEXTYPE (TYP ZŁOŻONY) | DEKLARACJA

- w XML Schema może być zdefiniowany na dwa sposoby
  - bezpośrednia deklaracja elementu złożonego przez nazwanie go
  - przypisanie elementowi złożonemu nazwy i atrybutu **type**, który odnosi się do nazwy
    - wtedy kilka elementów w schemacie może odwoływać się do tego samego typu złożonego

```
<xs:element name="produkt" type="rodzaj_produktu"/>

<xs:complexType name="rodzaj_produktu">
  <xs:attribute name="id" type="xs:positiveInteger"/>
</xs:complexType>
```

JĘZYK XML SCHEMA

ELEMENT COMPLEXTYPE (TYP ZŁOŻONY) | DEKLARACJA

- **pusty element złożony**
  - nie może mieć żadnej zawartości poza atrybutami

pusty element w **XML**

```
<produkt id="1345"/>

<produkt id="1345"></produkt>
```

definicja typu bez zawartości w **XML Schema**

```
<xs:element name="produkt">
  <xs:complexType>
    <xs:complexContent>
      <xs:restriction base="xs:integer">
        <xs:attribute name="id" type="xs:positiveInteger"/>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="produkt">
  <xs:complexType>
    <xs:attribute name="id" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
```

ELEMENT COMPLEXTYPE (TYP ZŁOŻONY) | DEKLARACJA

- **element złożony zawierający tylko inne elementy**

element **XML** "osoba" zawiera tylko inne elementy

```
<osoba>
  <imię>Jan</imię>
  <nazwisko>Kowalski</nazwisko>
</osoba>
```

definicja elementu "osoba" w **XML Schema**

```
<xs:element name="osoba">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="imię" type="xs:string"/>
      <xs:element name="nazwisko" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

JĘZYK XML SCHEMA

ELEMENT COMPLEXTYPE (TYP ZŁOŻONY) | DEKLARACJA

- **element złożony zawierający tylko tekst**
  - może zawierać tekst i atrybuty (zawartość prostą)
    - stąd dodaje się element *simpleContenet*

element **XML** "rozmiar_buta", który zawiera tylko tekst

```
<rozmiar_buta kraj="Polska">39</rozmiar_buta>
```

definicja elementu "rozmiar_buta" w **XML Schema**

```
<xs:element name="rozmiar_buta">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:integer">
        <xs:attribute name="kraj" type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

ELEMENT COMPLEXTYPE (TYP ZŁOŻONY) | DEKLARACJA

- **element złożony z mieszaną zawartością**
  - może zawierać atrybuty, elementy i tekst

element **XML** "list", który zawiera tekst i inne elementy

```
<list>
  Szanowny Panie<nazwisko>Kowalski</nazwisko>.
  Pana zamówienie<id_zamówienia>1032</id_zamówienia>
  zostanie zrealizowane<data_dostawy>2011-06-25</data_dostawy>
</list>
```

definicja "list" elementu w **XML Schema**

```
<xs:element name="list">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="nazwisko" type="xs:string"/>
      <xs:element name="id_zamówienia" type="xs:positiveInteger"/>
      <xs:element name="data_dostawy" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

JĘZYK XML SCHEMA

ELEMENT COMPLEXTYPE (TYP ZŁOŻONY) | DEKLARACJA

- do zdefiniowania nowych elementów prostych i złożonych (*simpleType* i *complexType*) można wykorzystać elementy już istniejące w schemacie
  - rozszerzyć je (**extension**) i dodać do nich nowe elementy

```
<xs:element name="pracownik" type="pełne_dane_osobowe"/>

<xs:complexType name="dane_osobowe">
  <xs:sequence>
    <xs:element name="imię" type="xs:string"/>
    <xs:element name="nazwisko" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="pełne_dane_osobowe">
  <xs:complexContent>
    <xs:extension base="dane_osobowe">
      <xs:sequence>
        <xs:element name="ulica" type="xs:string"/>
        <xs:element name="kod_pocztowy" type="xs:string"/>
        <xs:element name="miasto" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

WSKAŹNIKI

- kontrolują sposób używania elementów w dokumencie XML
  - **wskaźniki porządkowe**
  - **wskaźniki występowania**
  - **wskaźniki grupy**

JĘZYK XML SCHEMA

WSKAŹNIKI PORZĄDKOWE

- definiują kolejność elementów
  - **all**
    - elementy „dzieci” mogą pojawić się w dowolnej kolejności
    - każdy element „dziecko” może pojawić się tylko raz
  - **choice**
    - może wystąpić jeden albo więcej elementów „dziecko”
  - **sequence**
    - elementy „dzieci” muszą pojawić się w określonej kolejności

WSKAŹNIKI WYSTĘPOWANIA

- definiują częstość występowania elementów
  - **maxOccurs**
    - max ilość wystąpień elementu
  - **minOccurs**
    - min ilość wystąpień elementu

JĘZYK XML SCHEMA

WSKAŹNIKI GRUPY

- definiują powiązane zbiory elementów
  - **group**
    - nazwana grupa elementów
  - **attributeGroup**
    - nazwana grupa atrybutów

WSKAŹNIKI | PRZYKŁAD

```
<xs:group name="podstawowe_dane_osobowe">
  <xs:sequence>
    <xs:element name="imię" type="xs:string" maxOccurs="3"/>
    <xs:element name="nazwisko" type="xs:string"/>
    <xs:element name="data_urodzenia" type="xs:date" minOccurs="0"/>
  </xs:sequence>
</xs:group>

<xs:element name="osoba" type="dane_osobowe"/>

<xs:complexType name="dane_osobowe">
  <xs:sequence>
    <xs:group ref="podstawowe_dane_osobowe"/>
    <xs:element name="narodowość" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

- zdefiniowano grupę elementów "podstawowe_dane_osobowe" (wskaźnik *group*)
  - elementy w grupie muszą pojawić się dokładnie w podanej kolejności (wskaźnik *sequence*)
  - element "imię" może pojawić się maksymalnie 3 razy (wskaźnik *maxOccurs*)
  - element "data_urodzenia" jest opcjonalny (wskaźnik *minOccurs*)
- po zdefiniowaniu grupy, można się do niej odwołać w innej definicji (group *ref*="podstawowe_dane_osobowe")

JĘZYK XML SCHEMA

ELEMENT ZASTĘPOWANIA

- w XML Schema jeden element można zastąpić innym elementem
  - elementem zastępowania (**substitutionGroup**)
    - najpierw należy zadeklarować element główny (ang. *head*)
    - następnie pozostałe elementy
      - stanowiące zastępstwo dla elementu głównego

ELEMENT ZASTĘPOWANIA | PRZYKŁAD

```
<xs:element name="nazwisko" type="xs:string"/>
<xs:element name="pseudonim" substitutionGroup="nazwisko"/>

<xs:complexType name="muzyk">
  <xs:sequence>
    <xs:element ref="nazwisko"/>
  </xs:sequence>
</xs:complexType>
```

- element "nazwisko" może zostać zastąpiony elementem "pseudonim"
- poprawne dokumenty XML według powyższego XML Schema

```
<muzyk>
  <nazwisko>Smolik</nazwisko>
</muzyk>
```

lub

```
<muzyk>
  <pseudonim>Smolik</pseudonim>
</muzyk>
```

JĘZYK XML SCHEMA

ELEMENTY INCLUDE I IMPORT

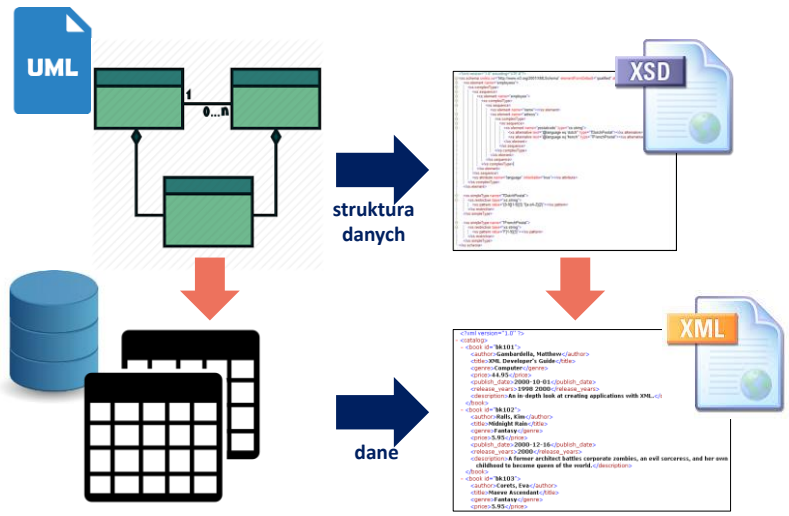
- pozwalają na dodanie do dokumentu XML Schema schematów
  - z tą samą docelową przestrzenią nazw
    - **include**
  - z różnymi docelowymi przestrzeniami nazw
    - **import**

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.w3schools.com/schema">

<xs:include schemaLocation="http://www.w3schools.com/schema/company.xsd"/>
<xs:import namespace=http://www.isotc211.org/2005/gco
           schemaLocation="http://www.isotc211.org/2005/gco/gco.xsd"/>

</xs:schema>
```

PODSUMOWANIE | XSD VS XML



DLACZEGO WARTO ZNAĆ XML?

- standard XML ze względu na
  - otwartość
  - niezależność od platformy sprzętowej
  - rozszerzalność i łatwość transformacji
  - zgodność z HTML

  znajduje zastosowanie w wielu obszarach
  - **opis zasobów**
    - pozwala na tworzenie deskryptorów zasobów sieciowych
    - nadaje się do opisu wiedzy w postaci ontologii
    - standard RDF (ang. *Resource Description Framework*) bazuje na XML oraz URI (odsyłaczach do zasobów)
    - WDSL (ang. *Web Services Description Language*) wykorzystuje XML do opisu punktów dostępu do usług sieciowych (Web Services)
  - **reprezentacja informacji semistrukturalnej**
    - XML nadaje się do zapisu dokumentów w postaci częściowo ustrukturalizowanej

[źródło: http://www.inzynieriawiedzy.pl/]

DLACZEGO WARTO ZNAĆ XML?

PODSUMOWANIE

- **multimedia**
  - bezpośredni zapis multimediów
    - np. SVG (ang. *Scalable Vector Graphics*) , uniwersalny format dwuwymiarowej grafiki wektorowej
  - sterowanie przetwarzaniem informacji multimedialnej
    - np. SMIL (ang. *Synchronized Multimedia Integration Language*)
- **specjalistyczne struktury danych**
  - tworzenie specjalistycznych struktur do przekazywania wiedzy z danej dziedziny naukowej, np.
    - MathML (ang. *Mathematical Markup Language*), do opisywania wzorów i symboli matematycznych
    - MusicXML, znacznikowy format prezentacji graficznej notacji muzycznej
    - CML (ang. *Chemistral Markup Language*), do opisu wyrażeń stosowanych w chemii
- **elektroniczna wymiana danych (EDI) i dokumentów**
  - ułatwia handel elektroniczny typu B2B (ang. *business to business*) oraz integrację oddzielnych systemów informatycznych
  - nowsze schematy ułatwiające wymianę danych między firmami
    - np. ebXML (ang. *electronic business XML*)

[źródło: http://www.inzynieriawiedzy.pl/]

DLACZEGO WARTO ZNAĆ XML?

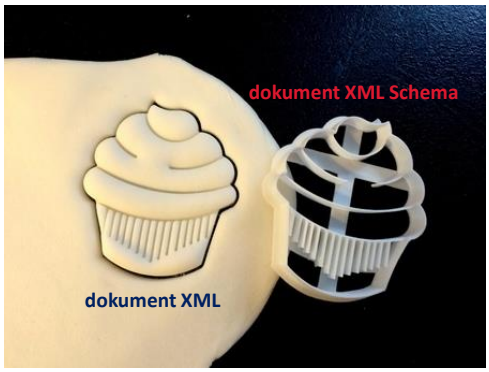
PODSUMOWANIE

- **protokoły komunikacyjne**
  - wykorzystywany w protokołach wymiany komunikatów
    - np. protokół zdalnego dostępu do obiektów SOAP (ang. *Simple Object Acces Protocol*) używa XML do kodowania wywołań
- **komunikacja w sferze publicznej**
  - elektroniczna komunikacja między obywatelami a jednostkami administracji publicznej oparta na standardzie XML
- **tworzenie stron internetowych**
  - generowanie dokumentów XML przy użyciu stylów CSS
  - stosowanie języka przekształceń XSLT, który pozwala na przetłumaczenie dokumentów z jednego formatu XML na dowolny inny format zgodny ze składnią XML, np. na
    - stronę WWW XHTML
    - wzór matematyczny MathML
    - zwykły HTML i czysty tekst

[źródło: http://www.inzynieriawiedzy.pl/]

DLACZEGO WARTO ZNAĆ XML?

- gramatyka **języka XML** i **XML Schema** stanowi podstawę dla gramatyki **języka GML**



NARZĘDZIA WSPOMAGAJĄCE TWORZENIE PLIKÓW XML I XSD

NARZĘDZIA DEDYKOWANE PLIKOM XML I XSD

- **edytor**
  - tworzenie i edycja dokumentów XML i XSD
- **parser**
  - analizowanie i sprawdzanie poprawności składni (struktury) dokumentów XML i XSD

- **walidator**
  - sprawdzanie poprawności składniowej plików XML i XSD
  - kontrola zgodności z oficjalną specyfikacją
    - zgodność plików XML i XSD ze specyfikacjami W3C
    - zgodność dokumentu XML ze strukturą zdefiniowaną w pliku XSD

NARZĘDZIA DEDYKOWANE PLIKOM XML I XSD

NARZĘDZIA WSPOMAGAJĄCE EDYCJĘ PLIKÓW XML I XSD | PRZYKŁADY

- **on-line**
  - **Tutorials Point**
    - *https://www.tutorialspoint.com/online_xml_editor.htm*
  - **Code Beautify**
    - *https://codebeautify.org/xmlviewer*
  - **XML Viewer**
    - *http://www.xmlviewer.org/*
  - **XML Formatter**
    - *https://www.freeformatter.com/xml-formatter.html*
  - **XSD/XML Schema Generator**
    - *https://www.freeformatter.com/xsd-generator.html*

NARZĘDZIA DEDYKOWANE PLIKOM XML I XSD

NARZĘDZIA WSPOMAGAJĄCE EDYCJĘ PLIKÓW XML I XSD | PRZYKŁADY

- **desktop**
  - darmowe
    - **Notepad ++**
      - *https://notepad-plus-plus.org/*
    - **EditPad Lite**
      - *https://www.editpadlite.com/*

  - komercyjne
    - **Altova XMLSpy**
      - *https://www.altova.com/xmlspy-xml-editor*
    - **Oxygen XML Editor**
      - *https://www.oxygenxml.com/*

NARZĘDZIA DEDYKOWANE PLIKOM XML I XSD

NARZĘDZIA WSPOMAGAJĄCE WALIDACJĘ PLIKÓW XML I XSD | PRZYKŁADY

- **on-line**
  - **XML Validator**
    - *https://www.w3schools.com/xml/xml_validator.asp*
  - **Truugo**
    - *http://www.truugo.com/xml_validator/*
  - **W3C XML Schema (XSD) Validation online**
    - *http://www.utilities-online.info/xsdvalidation/#.WdImnMZpEy4*
  - **XML Validator – XSD (XML Schema)**
    - *https://www.freeformatter.com/xml-validator-xsd.html*
  - **XML Validator Online**
    - *http://xmlvalidator.new-studio.org/*

NARZĘDZIA WSPOMAGAJĄCE WALIDACJĘ PLIKÓW XML I XSD | PRZYKŁADY

- **desktop**
  - darmowe
    - **Notepad ++**
      - *https://notepad-plus-plus.org/*
      - wtyczka *XML Tools*
    - **AltovaXML Community Edition 2013**
      - *http://www.softpedia.com/get/Internet/Other-Internet-Related/AltovaXML.shtml*
      - brak interfejsu graficznego, obsługa tylko z poziomu wiersza poleceń
    - **XML Copy Editor**
      - *http://xml-copy-editor.sourceforge.net/*
  - komercyjne
    - **Altova XMLSpy**
      - *https://www.altova.com/xmlspy-xml-editor*
    - **Oxygen XML Editor**
      - *https://www.oxygenxml.com/*

ĆWICZENIE 1: XSD & XML I

- **pliki XSD & XML**
  - przejrzeć pliki XSD i XML dostępne w folderze *Jaskinie*
    - *DANE_XML\XSD&XML*
      - *Jaskinie_[1-4].xsd*
      - *Jaskinie_[1-4]g.xml*
  - zwrócić uwagę na różne konstrukcje zawarte w plikach XSD
  - wykorzystać aplikację *Notepad++*
    - wtyczka *XML Tools* → funkcja *Validate now*
  - wykorzystać aplikację *XML Copy Editor*
    - *Validate*
    - *XML → Validate → DTD/XMLSchema*

ĆWICZENIE 2: XML I

- **walidacja plików XML**
  - zdiagnozować pliki XML dostępne w katalogu *Jaskinie*
    - *DANE_XML\XSD&XML*
      - *Jaskinie_[1-4]b.xml*
  - poprawić błędy wykryte w plikach XML tak, aby pliki pomyślnie przechodziły proces walidacji odpowiednimi schematami (plikami XSD)
    - *Jaskinie_[1-4]b.xsd*
  - wykorzystać aplikację *Notepad++*
    - wtyczka *XML Tools* → funkcja *Validate now*
  - wykorzystać aplikację *XML Copy Editor*
    - *Validate*
    - *XML → Validate → DTD/XMLSchema*
  - wykorzystać aplikację *AltovaXML Community Edition 2013*
    - *DANE_XML\XSD&XML\Walidacja*

JĘZYK XML SCHEMA & XML

ĆWICZENIE 3: XML II

- **walidacja pliku XML**
  - zdiagnozować plik XML z metadanymi *MapaGeologicznaBałtyku.xml*
    - *DANE_XML\XSD&XML\Metadane*
  - poprawić błędy wykryte w pliku XML tak, aby plik pomyślnie przechodził proces walidacji schematem (plikiem XSD)
    - *gmd.xsd*
  - wykorzystać aplikację *Notepad++*
    - wtyczka *XML Tools* → funkcja *Validate now*
  - wykorzystać aplikację *XML Copy Editor*
    - *Validate*
    - *XML → Validate → DTD/XMLSchema*

ĆWICZENIE 4: XSD & XML II

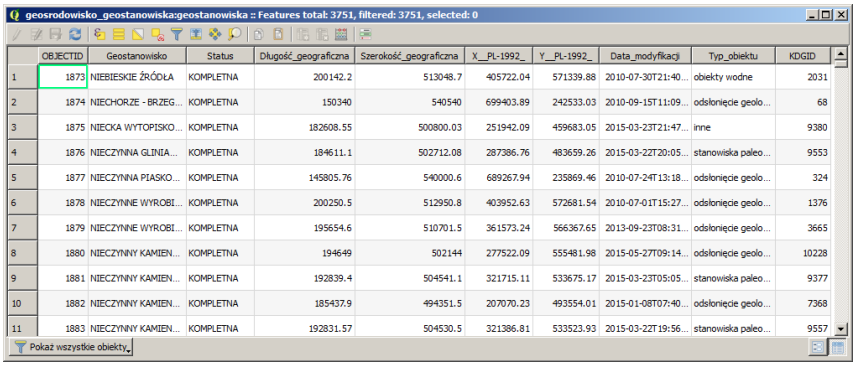
- **tworzenie plików XSD & XML**
  - w aplikacji *QGIS* wczytać usługę WFS *geostanowiska* Centralnej Bazy Danych Geologicznych
    - adres usługi dostępny w pliku *WFS.txt*
      - *DANE_XML\XSD&XML\GeoStanowiska*
      - *http://cbdgmapa.pgi.gov.pl/arcgis/services/geosrodowisko/geostanowiska/MapServer/WFSServer*
  - na podstawie zawartości tabeli atrybutów opracować dokument XSD
    - dla elementu *Typ_obiektu* zastosować ograniczenie *enumeration*
  - na podstawie opracowanego pliku XSD przygotować próbkę danych XML

  - wykorzystać aplikację *Notepad++*
    - wtyczka *XML Tools* → funkcja *Validate now*
  - wykorzystać aplikację *XML Copy Editor*
    - *Validate*
    - *XML* → *Validate* → *DTD/XMLSchema*

JĘZYK XML SCHEMA & XML

ĆWICZENIE 4: XSD & XML II

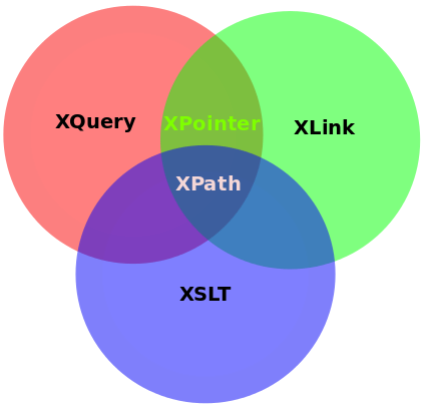
- **tworzenie plików XSD & XML**



	OBJECTID	Geostanowisko	Status	Długość_geograficzna	Szerokość_geograficzna	X__PL-1992_	Y__PL-1992_	Data_modyfikacji	Typ_obiektu	KDGID
1	1873	NIEBIESKIE ŹRÓDŁA	KOMPLETNA	200142.2	513048.7	405722.04	571339.88	2010-07-30T21:40...	obiekty wodne	2031
2	1874	NIECHORZE - BRZEG...	KOMPLETNA	150340	540540	699403.89	242533.03	2010-09-15T11:09...	odsłonięcie geolo...	68
3	1875	NIECKA WYTOPISKO...	KOMPLETNA	182608.55	500800.03	251942.09	459683.05	2015-03-23T21:47...	inne	9380
4	1876	NIECZYNNA GLINIA...	KOMPLETNA	184611.1	502712.08	287386.76	483659.26	2015-03-22T20:05...	stanowiska paleo...	9553
5	1877	NIECZYNNA PIASKO...	KOMPLETNA	145805.76	540000.6	689267.94	235869.46	2010-07-24T13:18...	odsłonięcie geolo...	324
6	1878	NIECZYNNE WYROBI...	KOMPLETNA	200250.5	512950.8	403952.63	572681.54	2010-07-01T15:27...	odsłonięcie geolo...	1376
7	1879	NIECZYNNE WYROBI...	KOMPLETNA	195654.6	510701.5	361573.24	566367.65	2013-09-23T08:31...	odsłonięcie geolo...	3665
8	1880	NIECZYNNY KAMIEN...	KOMPLETNA	194649	502144	277522.09	555481.98	2015-05-27T09:14...	odsłonięcie geolo...	10228
9	1881	NIECZYNNY KAMIEN...	KOMPLETNA	192839.4	504541.1	321715.11	533675.17	2015-03-23T05:05...	stanowiska paleo...	9377
10	1882	NIECZYNNY KAMIEN...	KOMPLETNA	185437.9	494351.5	207070.23	493554.01	2015-01-08T07:40...	odsłonięcie geolo...	7368
11	1883	NIECZYNNY KAMIEN...	KOMPLETNA	192831.57	504530.5	321386.81	533523.93	2015-03-22T19:56...	stanowiska paleo...	9557

PRZYPOMNIENIE

RODZINA JĘZYKÓW XML



RODZINA JĘZYKÓW XML

- **Xlink**
  - tworzenie hiperłączy w dokumentach XML
- **XPointer**
  - odwoływanie się do określonych części dokumentu XML
  - wykorzystuje wyrażenia XPath do nawigacji w ramach dokumentu XML
- **XPath**
  - definiowanie ścieżek do nawigacji w dokumentach XML
  - podstawowy element dla XSLT i Xquery
- **XSLT**
  - przekształcanie dokumentu XML na dokument HTML
  - wykorzystuje XPath do odnalezienia informacji w dokumencie XML
- **XQuery**
  - język zapytań dla XML (jak SQL dla bazy danych)

JĘZYK XLINK

- ang. ***X**ML **Linking** Language*
- „język łączy” w dokumentach XML
- standard opracowany przez W3C

- umożliwia tworzenie łączy URI w dokumentach XML
  - nie jest wspierany przez przeglądarki internetowe

XLINK | ZASADY SKŁADNI

- dowolny element w dokumencie XML może pełnić rolę łącza
- należy zadeklarować przestrzeń nazw Xlink (dostęp do właściwości Xlink)
  - *http://www.w3.org/1999/xlink*

```
<strony_www xmlns:xlink="http://www.w3.org/1999/xlink">
  <strona_domowa xlink:type="simple"
                 xlink:href="https://www.mos.gov.pl/">Ministerstwo Środowiska</strona_domowa>
  <strona_archiwalna xlink:type="simple"
                 xlink:href="http://archiwum.mos.gov.pl/">Strona archiwalna MŚ</strona_archiwalna>
</strony_www>
```

- atrybuty *xlink:type* oraz *xlink:href* elementu <strona_domowa> i <strona_archiwalna> pochodzą z przestrzeni nazw XLink
  - *xlink:type="simple"* określa proste łącze (np. „naciśnij tu, żeby wejść”)
  - *xlink:href* określa adres URL łącza

JĘZYK XLINK

XLINK | PRZYKŁAD

```
<strona_domowa xmlns:xlink="http://www.w3.org/1999/xlink">
  <logo xlink:type="simple"
        xlink:href="https://www.mos.gov.pl/fileadmin/templates/Resources/Public/Images/logo.png"
        xlink:show="new">Ministerstwo Środowiska</logo>
</strona_domowa>
```

- *xmlns:xlink="http://www.w3.org/1999/xlink"*
  - przestrzeń nazw XLink
- *xlink:type="simple"*
  - łącze proste (jak w języku HTML)
- *xlink:href*
  - określa adres URL łącza (tu: rysunek)
- *xlink:show="new"*
  - określa, że łącze otworzy się w nowym oknie

XLINK | MOŻLIWOŚCI

Atrybut XML	Wartość	Opis
xlink:actuate	onLoad onRequest other none	określa, kiedy łącze ma zadziałać, np. ▪ onLoad – zasób źródłowy („linkowany") zostanie załadowany w momencie załadowania dokumentu XML ▪ onRequest – zasób źródłowy zostanie załadowany w momencie kliknięcia na łącze
xlink:href	URL	określa adres URL łącza
xlink:show	embed new replace other none	określa, gdzie łącze zostanie otwarte (wartość domyślna to „replace" – łącze zastąpi aktualne okno)
xlink:type	simple extended locator arc resource title none	określa rodzaj łącza

JĘZYK XPOINTER

- ang. ***X**ML **Pointer** Language*
- „język wskaźników" w dokumentach XML
- standard opracowany przez W3C

- umożliwia wskazywanie określonych części dokumentu XML
  - nie jest wspierany przez przeglądarki internetowe

XPOINTER VS XLINK | ZASADY SKŁADNI

- **XPointer**
  - odwoływanie się do określonych części dokumentu XML
    - *xlink:href="https://mos.gov.pl/ stronyMS.xml**#xpointer(id(domMS))**"*
  - metoda skrócona – użycie wartości atrybutu *id*
    - *xlink:href="https://mos.gov.pl/stronyMS.xml**#domMS**"*

- **XLink**
  - odwoływania się do całego dokumentu XML
    - *xlink:href="https://mos.gov.pl/stronyMS.xml*

XPOINTER | PRZYKŁAD

```
<?xml version="1.0" encoding="UTF-8"?>
<strony_www>

  <strona tytuł="Ministerstwo Środowiska" id="domMS">
    <adres_URL>https://www.mos.gov.pl/</adres_URL>
    <ostatnia_modyfikacja>23-10-2017</ostatnia_modyfikacja>
  </strona>

  <strona tytuł="Strona archiwalna MŚ" id="archMS">
    <adres_URL>http://archiwum.mos.gov.pl/</adres_URL>
    <ostatnia_modyfikacja>27-10-2016</ostatnia_modyfikacja>
  </strona>

</strony_www>
```

docelowy dokument XML " stronyMS.xml "
(do jego elementów będzie odwołanie
z innego dokumentu XML)

- element <strona> posiada unikalny identyfikator
  - atrybut *id*

XPOINTER | PRZYKŁAD

```
<?xml version="1.0" encoding="UTF-8"?>

<wykaz_stron xmlns:xlink="http://www.w3.org/1999/xlink">

  <strona>
    <opis>Strona domowa Ministerstwa Środowiska</opis>
    <adres xlink:type="simple" xlink:href="https://mos.gov.pl/stronyMS.xml#domMS">Ministerstwo Środowiska</adres>
  </strona>

  <strona>
    <opis>Strona archiwalna Ministerstwa Środowiska</opis>
    <adres xlink:type="simple" xlink:href="https://mos.gov.pl/stronyMS.xml#archMS">Ministerstwo Środowiska</adres>
  </strona>

</wykaz_stron>
```

dokument XML zawierający łącza ze wskazaniem
na konkretne elementy innego dokumentu XML ("stronyMS.xml ")

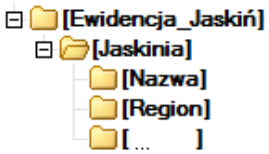
JĘZYK XPATH

- ang. ***X**ML **Path** Language*
- język ścieżek w dokumentach XML
- standard opracowany przez W3C

- podstawowy składnik języków **XSLT** i **XQuery**
- umożliwia wybór określonych elementów dokumentu XML

XPATH

- określa ścieżki dostępu do poszczególnych elementów w dokumencie XML



- zawiera ponad 200 wbudowanych funkcji
  - np. do przetwarzania wartości tekstowych, liczbowych, węzłów, sekwencji
- może być stosowany w innych językach
  - np. JavaScript, Java, XML Schema, PHP, Python, C, C++

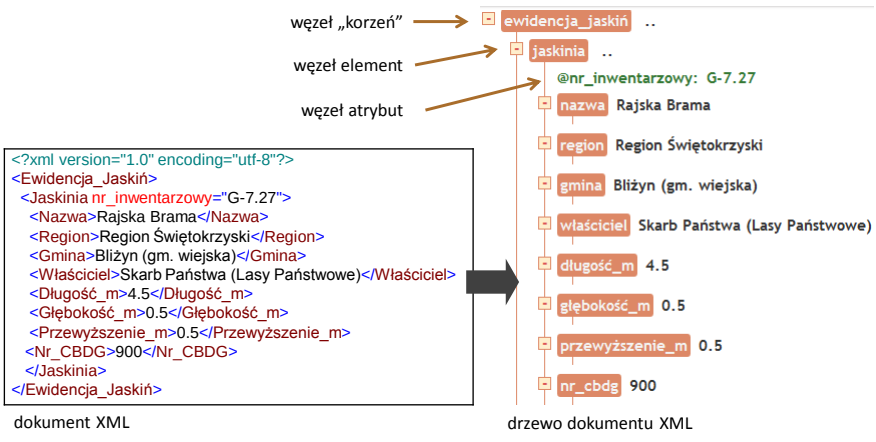
XPATH | WĘZŁY

- dokument XML jest traktowany jak „drzewo węzłów” („choinka”)

- wyróżnia się 7 rodzajów węzłów
  - element
  - atrybut
  - tekst
  - przestrzeń nazw
  - instrukcja przetwarzania
  - komentarz
  - dokument

- korzeń (ang. *root*)
  - węzeł położony najwyżej w „drzewie”

XPATH | WĘZŁY | PRZYKŁAD




```
<?xml version="1.0" encoding="utf-8"?>
<Ewidencja_Jaskiń>
 <Jaskinia nr_inwentarzowy="G-7.27">
  <Nazwa>Rajska Brama</Nazwa>
  <Region>Region Świętokrzyski</Region>
  <Gmina>Bliżyn (gm. wiejska)</Gmina>
  <Właściciel>Skarb Państwa (Lasy Państwowe)</Właściciel>
  <Długość_m>4.5</Długość_m>
  <Głębokość_m>0.5</Głębokość_m>
  <Przewyższenie_m>0.5</Przewyższenie_m>
  <Nr_CBDG>900</Nr_CBDG>
 </Jaskinia>
</Ewidencja_Jaskiń>
```

dokument XML

drzewo dokumentu XML

XPATH | WARTOŚĆ ATOMOWA | POZYCJA

- wartość atomowa
  - węzeł bez „dzieci” i bez „rodziców”

```
"G-7.27"
Rajska Brama
4.5
```

- pozycja
  - wartość atomowa
  - węzeł

XPATH | RELACJE MIĘDZY WĘZŁAMI | RODZICE

- każdy element i atrybut ma 1 „rodzica”

```
<Jaskinia nr_inwentarzowy="G-7.27">
  <Nazwa>Rajska Brama</Nazwa>
  <Region>Region Świętokrzyski</Region>
  <Długość_m>4.5</Długość_m>
  <Głębokość_m>0.5</Głębokość_m>
</Jaskinia>
```

- element <Jaskinia> jest „rodzicem” dla elementów: <Nazwa>, <Region>, <Długość_m>, <Głębokość_m>

JĘZYK XPATH

XPATH | RELACJE MIĘDZY WĘZŁAMI | DZIECI

- elementy węzły mogą mieć 0, 1 lub wiele „dzieci”

```
<Jaskinia nr_inwentarzowy="G-7.27">
  <Nazwa>Rajska Brama</Nazwa>
  <Region>Region Świętokrzyski</Region>
  <Długość_m>4.5</Długość_m>
  <Głębokość_m>0.5</Głębokość_m>
</Jaskinia>
```

- elementy <Nazwa>, <Region>, <Długość_m>, <Głębokość_m> są „dziećmi” elementu <Jaskinia>

XPATH | RELACJE MIĘDZY WĘZŁAMI | RODZEŃSTWO

- węzły, które mają tego samego „rodzica”

```
<Jaskinia nr_inwentarzowy="G-7.27">
  <Nazwa>Rajska Brama</Nazwa>
  <Region>Region Świętokrzyski</Region>
  <Długość_m>4.5</Długość_m>
  <Głębokość_m>0.5</Głębokość_m>
</Jaskinia>
```

- elementy <Nazwa>, <Region>, <Długość_m>, <Głębokość_m> są „rodzeństwem”

JĘZYK XPATH

XPATH | RELACJE MIĘDZY WĘZŁAMI | PRZODKOWIE

- „rodzic” węzła, „rodzic rodzica” węzła, itd.

```
<Ewidencja_Jaskiń>
  <Jaskinia nr_inwentarzowy="G-7.27">
    <Nazwa>Rajska Brama</Nazwa>
    <Region>Region Świętokrzyski</Region>
    <Gmina>Bliżyn (gm. wiejska)</Gmina>
    <Właściciel>Skarb Państwa (Lasy Państwowe)</Właściciel>
    <Długość_m>4.5</Długość_m>
    <Głębokość_m>0.5</Głębokość_m>
    <Przewyższenie_m>0.5</Przewyższenie_m>
    <Nr_CBDG>900</Nr_CBDG>
  </Jaskinia>
</Ewidencja_Jaskiń>
```

- przodkami dla elementu <Nazwa> są elementy: <Jaskinia> oraz <Ewidencja_Jaskiń>

XPATH | RELACJE MIĘDZY WĘZŁAMI | POTOMKOWIE

- „dzieci” węzła, „dzieci dzieci” węzła, itd.

```
<Ewidencja_Jaskiń>
  <Jaskinia nr_inwentarzowy="G-7.27">
    <Nazwa>Rajska Brama</Nazwa>
    <Region>Region Świętokrzyski</Region>
    <Gmina>Bliżyn (gm. wiejska)</Gmina>
    <Właściciel>Skarb Państwa (Lasy Państwowe)</Właściciel>
    <Długość_m>4.5</Długość_m>
    <Głębokość_m>0.5</Głębokość_m>
    <Przewyższenie_m>0.5</Przewyższenie_m>
    <Nr_CBDG>900</Nr_CBDG>
  </Jaskinia>
</Ewidencja_Jaskiń>
```

- potomkami elementu <Ewidencja_Jaskiń> są elementy: <Jaskinia>, <Nazwa>, <Region>, <Gmina>, <Właściciel>, <Długość_m>, <Głębokość_m>, <Przewyższenie_m>, <Nr_CBDG>

JĘZYK XPATH

XPATH | WYBÓR WĘZŁA

- „wyrażenia ścieżek” pozwalają na wybór węzłów lub zbioru węzłów w dokumencie XML
- węzeł zostaje wybrany poprzez śledzenie ścieżki lub poszczególnych jej kroków

Wyrażenie	Opis
nazwa_węzła	wybór wszystkich węzłów o nazwie „nazwa_węzła”
/	wybór z węzła „korzeń”
//	wybór węzłów z węzła bieżącego, niezależnie od ich położenia w drzewie dokumentu XML
.	wybór węzła bieżącego
..	wybór „rodzica” węzła bieżącego
@	wybór atrybutów

XPATH | WYBÓR WĘZŁA | PRZYKŁAD

```
<Ewidencja_Jaskiń>
  <Jaskinia nr_inwentarzowy="G-7.27">
    <Nazwa>Rajska Brama</Nazwa>
    <Region>Region Świętokrzyski</Region>
    <Długość_m>4.5</Długość_m>
    <Głębokość_m>0.5</Głębokość_m>
  </Jaskinia>
  <Jaskinia nr_inwentarzowy="T.E-10.03">
    <Nazwa>Dziura nad Studnią</Nazwa>
    <Region>Tatry</Region>
    <Długość_m>20</Długość_m>
    <Głębokość_m>6.2</Głębokość_m>
  </Jaskinia>
</Ewidencja_Jaskiń>
```

Ścieżka	Wynik
Ewidencja_Jaskiń	wybór wszystkich węzłów o nazwie „Ewidencja_Jaskiń”
/Ewidencja_Jaskiń	wybór elementu „korzeń” o nazwie „Ewidencja_Jaskiń” (ścieżka rozpoczynająca się od znaku ukośnika (/) jest ścieżką bezwzględną do elementu)
/Ewidencja_Jaskiń/Jaskinia	wybór wszystkich elementów „Jaskinia”, które są „dziećmi” elementu „Ewidencja_Jaskiń”
//Jaskinia	wybór wszystkich elementów „Jaskinia”, niezależnie od ich położenia w drzewie dokumentu XML
/Ewidencja_Jaskiń//Jaskinia	wybór wszystkich elementów „Jaskinia”, które są „potomkami” elementu „Ewidencja_Jaskiń”, niezależnie od ich położenia w drzewie dokumentu XML
//@nr_inwentarzowy	wybór wszystkich atrybutów o nazwie „nr_inwentarzowy”

JĘZYK XPATH

XPATH | PREDYKAT

- stosowany do wyszukiwania określonego węzła lub węzła, który zawiera określoną właściwość
- zawsze osadzony w nawiasach kwadratowych
  - np. /Ewidencja_Jaskiń/Jaskinia*[Głębokość_m>2.0]*

XPATH | PREDYKAT | PRZYKŁAD

```
<Ewidencja_Jaskiń>
  <Jaskinia nr_inwentarzowy="G-7.27">
    <Nazwa>Rajska Brama</Nazwa>
    <Region>Region Świętokrzyski</Region>
    <Długość_m>4.5</Długość_m>
    <Głębokość_m>0.5</Głębokość_m>
  </Jaskinia>
  <Jaskinia nr_inwentarzowy="T.E-10.03">
    <Nazwa>Dziura nad Studnią</Nazwa>
    <Region>Tatry</Region>
    <Długość_m>20</Długość_m>
    <Głębokość_m>6.2</Głębokość_m>
  </Jaskinia>
</Ewidencja_Jaskiń>
```

Ścieżka	Wynik
/Ewidencja_Jaskiń/Jaskinia[1]	wybór pierwszego elementu „Jaskinia”, który jest „dzieckiem” elementu „Ewidencja_Jaskiń”
/Ewidencja_Jaskiń/Jaskinia[last()]	wybór ostatniego elementu „Jaskinia”, który jest „dzieckiem” elementu „Ewidencja_Jaskiń”
/Ewidencja_Jaskiń/Jaskinia[last()-1]	wybór ostatniego, ale jednego elementu „Jaskinia”, który jest „dzieckiem” elementu „Ewidencja_Jaskiń”
/Ewidencja_Jaskiń/Jaskinia[position()<3]	wybór pierwszych dwóch elementów „Jaskinia”, które są „dziećmi” elementu „Ewidencja_Jaskiń”
//Jaskinia[@nr_inwentarzowy]	wybór wszystkich elementów „Jaskinia”, które posiadają atrybut „nr_inwentarzowy”
//Jaskinia[@nr_inwentarzowy='G-7.27']	wybór wszystkich elementów „Jaskinia”, które posiadają atrybut „nr_inwentarzowy” o wartości „G-7.27”
/Ewidencja_Jaskiń/Jaskinia[Głębokość_m>2.0]	wybór wszystkich elementów „Jaskinia” z elementu „Ewidencja_Jaskiń”, które posiadają element „Głębokość_m” o wartości większej niż „2.0”
/Ewidencja_Jaskiń/Jaskinia[Głębokość_m>2.0]/Nazwa	wybór wszystkich elementów „Nazwa” z elementów „Jaskinia” w elemencie „Ewidencja_Jaskiń”, które posiadają element „Głębokość_m” o wartości „> 2.0”

JĘZYK XPATH

XPATH | WYBÓR NIEWIADOMEGO WĘZŁA

- do selekcji niewiadomych węzłów XML mogą być stosowane znaki wieloznaczne

Znak wieloznaczny	Opis
*****	oznacza dowolny element „węzeł”
@*	oznacza dowolny atrybut
node()	oznacza dowolny węzeł, dowolnego rodzaju

XPATH | WYBÓR NIEWIADOMEGO WĘZŁA | PRZYKŁAD

```
<Ewidencja_Jaskiń>
  <Jaskinia nr_inwentarzowy="G-7.27">
    <Nazwa>Rajska Brama</Nazwa>
    <Region>Region Świętokrzyski</Region>
    <Długość_m>4.5</Długość_m>
    <Głębokość_m>0.5</Głębokość_m>
  </Jaskinia>
  <Jaskinia nr_inwentarzowy="T.E-10.03">
    <Nazwa>Dziura nad Studnią</Nazwa>
    <Region>Tatry</Region>
    <Długość_m>20</Długość_m>
    <Głębokość_m>6.2</Głębokość_m>
  </Jaskinia>
</Ewidencja_Jaskiń>
```

Ścieżka	Wynik
/Ewidencja_Jaskiń/*	wybór wszystkich węzłów „dzieci" węzła „Ewidencja_Jaskiń"
//*	wybór wszystkich elementów (węzłów) w dokumencie
//Jaskinia[@*]	wybór wszystkich elementów „Jaskinia", które mają przynajmniej jeden dowolny atrybut

JĘZYK XPATH

XPATH | WYBÓR KILKU ŚCIEŻEK | PRZYKŁAD

- do selekcji wielu ścieżek stosuje się operator „|"

```
<Ewidencja_Jaskiń>
  <Jaskinia nr_inwentarzowy="G-7.27">
    <Nazwa>Rajska Brama</Nazwa>
    <Region>Region Świętokrzyski</Region>
    <Długość_m>4.5</Długość_m>
    <Głębokość_m>0.5</Głębokość_m>
  </Jaskinia>
  <Jaskinia nr_inwentarzowy="T.E-10.03">
    <Nazwa>Dziura nad Studnią</Nazwa>
    <Region>Tatry</Region>
    <Długość_m>20</Długość_m>
    <Głębokość_m>6.2</Głębokość_m>
  </Jaskinia>
</Ewidencja_Jaskiń>
```

Ścieżka	Wynik
//Jaskinia/Nazwa | //Jaskinia/Region	wybór wszystkich elementów „Nazwa" i „Region" spośród wszystkich elementów „Jaskinia"
//Nazwa | //Region	wybór wszystkich elementów „Nazwa" i „Region" w dokumencie XML
/Ewidencja_Jaskiń/Jaskinia/Nazwa | //Region	wybór wszystkich elementów „Nazwa" zawartych w elemencie „Jaskinia" elementu „Ewidencja_Jaskiń" i wszystkich elementów „Region" w dokumencie XML

XPATH | OŚ

- definiuje zestaw węzłów „pokrewnych” dla węzła bieżącego

Oś	Wynik
ancestor	wybór wszystkich przodków („rodziców”, „dziadków”, itd.) węzła bieżącego
ancestor-or-self	wybór wszystkich przodków („rodziców”, „dziadków”, itd.) węzła bieżącego i samego węzła bieżącego
attribute	wybór wszystkich atrybutów węzła bieżącego
child	wybór wszystkich dzieci węzła bieżącego
descendant	wybór wszystkich potomków („dzieci”, „wnuków”, itd.) węzła bieżącego
descendant-or-self	wybór wszystkich potomków („dzieci”, „wnuków”, itd.) węzła bieżącego i samego węzła bieżącego
following	wybór wszystkiego w dokumencie XML po znaczniku zamykającym węzeł bieżący, z wyjątkiem „przodków”, atrybutów i węzłów przestrzeni nazw
following-sibling	wybór całego „rodzeństwa” po węźle bieżącym
namespace	wybór wszystkich węzłów przestrzeni nazw węzła bieżącego
parent	wybór „rodzica” węzła bieżącego
preceding	wybór wszystkich węzłów, które znajdują się przed węzłem bieżącym w dokumencie XML
preceding-sibling	wybór całego „rodzeństwa” przed węzłem bieżącym
self	wybór węzła bieżącego

XPATH | ŚCIEŻKA LOKALIZACJI

- może być
  - bezwzględna (całkowita)
    - zaczyna się od znaku ukośnika („slash”, /)
  - względna

  - w obu przypadkach składa się z 1 lub kilku kroków oddzielonych znakiem ukośnika
    - ścieżka bezwzględna
      - */krok/krok/…*
    - ścieżka względna
      - *krok/krok/…*

XPATH | ŚCIEŻKA LOKALIZACJI I KROK

- każdy krok jest określany względem węzłów w bieżącym zestawie węzłów
- krok składa się z
  - osi
    - definiuje drzewo relacji między wybranymi węzłami a węzłem bieżącym
  - testu węzła
    - identyfikuje węzeł w ramach osi
  - 0 lub kilku predykatów
    - dalsze zawężenie wybranego zestawu węzłów
    - *oś::węzeł[predykat]*

XPATH | OŚ I ŚCIEŻKA LOKALIZACJI | PRZYKŁAD

```
<Ewidencja_Jaskiń>
  <Jaskinia nr_inwentarzowy="G-7.27">
    <Nazwa>Rajska Brama</Nazwa>
    <Region>Region Świętokrzyski</Region>
    <Długość_m>4.5</Długość_m>
    <Głębokość_m>0.5</Głębokość_m>
  </Jaskinia>
  <Jaskinia nr_inwentarzowy="T.E-10.03">
    <Nazwa>Dziura nad Studnią</Nazwa>
    <Region>Tatry</Region>
    <Długość_m>20</Długość_m>
    <Głębokość_m>6.2</Głębokość_m>
  </Jaskinia>
</Ewidencja_Jaskiń>
```

Przykład	Wynik
child::Jaskinia	wybór wszystkich węzłów „Jaskinia", które są „dziećmi" węzła bieżącego
attribute::nr_inwentarzowy	wybór atrybutu „nr_inwentarzowy" węzła bieżącego
child::*	wybór wszystkich elementów „dzieci" węzła bieżącego
attribute::*	wybór wszystkich atrybutów węzła bieżącego
child::text()	wybór wszystkich węzłów testowych „dzieci" węzła bieżącego
child::node()	wybór wszystkich „dzieci" węzła bieżącego
descendant::Jaskinia	wybór wszystkich „potomków" węzła bieżącego „Jaskinia"
ancestor::Jaskinia	wybór wszystkich „przodków" węzła bieżącego „Jaskinia"
ancestor-or-self::Jaskinia	wybór wszystkich „przodków" węzła bieżącego i samego węzła bieżącego, jeżeli jest to węzeł „Jaskinia"
child::*/child::Region	wybór wszystkich „wnuków" („dzieci dzieci") „Region" węzła bieżącego

XPATH | OPERATORY

- wyrażenie XPath może zwracać
  - zestaw węzłów
  - łańcuch znaków
  - liczbę
  - wartość boolowską (0 lub 1)

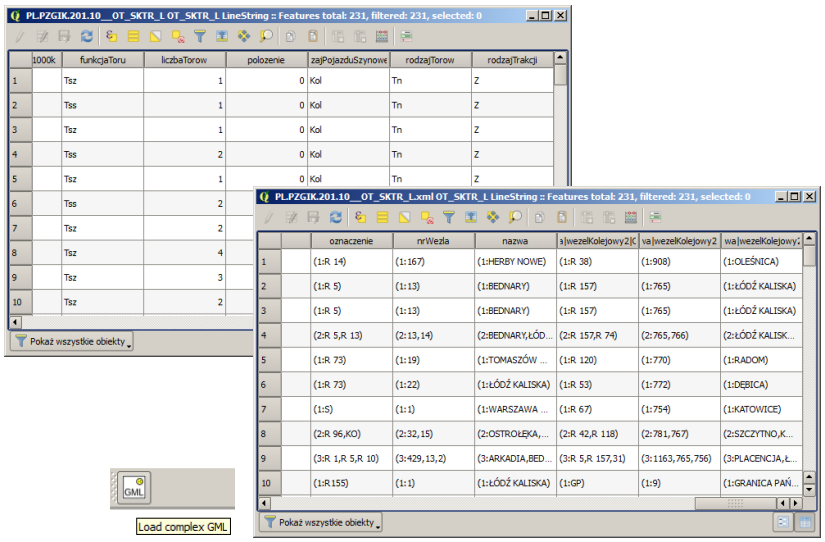
Operator	Opis	Przykład
|	przetwarzanie dwóch zestawów węzłów	//Nazwa | //Region
+	dodawanie	5 + 3
-	odejmowanie	5 - 3
*****	mnożenie	5 * 3
div	dzielenie	9 div 3
=	równe	=9.80
!=	różne	Głębokość_m!=11.20
<	mniejszy od	Głębokość_m <11.20
<=	mniejszy lub równy	Głębokość_m <=11.20
>	większy od	Głębokość_m >11.20
>=	większy lub równy	Głębokość_m >=11.20
or	lub	Głębokość_m =11.20 or Głębokość_m =15.20
and	i	Głębokość_m >11.00 and Głębokość_m <11.20
mod	modulo (reszta z dzielenia)	5 mod 3

ĆWICZENIE 5: XLINK, XPOINTER

- **XLink, XPointer**
  - plik *PL.PZGIK.201.10__OT_SKTR_L.xml* (tor, zespół torów) otworzyć w programie *Notepad++* oraz *QGIS*
    - *DANE_XML\XLink*
    - zwrócić uwagę na atrybut *xlink:href*
  - korzystając z wtyczki *GML Loader* ponownie dodać ten sam plik
    - zaobserwować różnicę
  - ↳ tabela atrybutów
    - brak *xlink:href* (ładowanie danych bez użycia wtyczki)
    - *xlink:href resolved* (ładowanie danych z wtyczką)

ĆWICZENIE 5: XLINK, XPOINTER

- **XLink, XPointer**



ĆWICZENIE 6: XPATH I

- **XPath**
  - dla danych zapisanych w pliku *Jaskinie.xml* (*DANE_XML\XPath*) podać wyrażenia XPath umożliwiające wybór poniższych elementów

Ścieżka	Pozycja kursora
	Ewidencja_Jaskin
	Jaskinia
	Region
	Glebokosc_m

  - wykorzystać aplikację *Notepad++*
    - wtyczka *XML Tools* → funkcja *Current XML Path*
  - wykorzystać aplikację *XML Copy Editor*
    - *XML* → *Copy The Current XPath*

ĆWICZENIE 6: XPATH I

- **XPath**

Ścieżka	Pozycja kursora
/Ewidencja_Jaskin	Ewidencja_Jaskin
/Ewidencja_Jaskin/Jaskinia	Jaskinia
/Ewidencja_Jaskin/Jaskinia/Region	Region
/Ewidencja_Jaskin/Jaskinia/Glebokosc_m	Glebokosc_m

ĆWICZENIE 7: XPATH II

- **XPath**
  - dla danych zapisanych w pliku *Jaskinie.xml* (*DANE_XML\XPath*) przetestować wyrażenia XPath dostępne w pliku *XPath.txt* (*DANE_XML\XPath*)

Ścieżka	Wynik
/Ewidencja_Jaskin/Jaskinia	
//@nr_inwentarzowy	
/Ewidencja_Jaskin/Jaskinia[last()-1]	
/Ewidencja_Jaskin/Jaskinia[Glebokosc_m>2.0]/Nazwa	
//Jaskinia[@*]	
/Ewidencja_Jaskin/Jaskinia/Nazwa | //Region	

  - wykorzystać aplikację *Notepad++*
    - wtyczka *XML Tools* → funkcja *Evaluate XPath expression*
  - wykorzystać aplikację *XML Copy Editor*
    - *XML* → *Evaluate XPath…*

ĆWICZENIE 7: XPATH II

- **XPath**

Ścieżka	Wynik
/Ewidencja_Jaskin/Jaskinia	nr_inwentarzowy="G-7.27" nr_inwentarzowy="T.E-10.03" nr_inwentarzowy="N-2.62"
//@nr_inwentarzowy	G-7.27 T.E-10.03 N-2.62
/Ewidencja_Jaskin/Jaskinia[last()-1]	nr_inwentarzowy="T.E-10.03"
/Ewidencja_Jaskin/Jaskinia[Glebokosc_m>2.0]/Nazwa	Dziura nad Studnia
//Jaskinia[@*]	nr_inwentarzowy="G-7.27" nr_inwentarzowy="T.E-10.03" nr_inwentarzowy="N-2.62"
/Ewidencja_Jaskin/Jaskinia/Nazwa | //Region	Rajska Brama Swietokrzyski Dziura nad Studnia Tatry Jaskinia Polkolista Niecka Nidzianska

JĘZYK XSLT

- ang. *e**X**tensible **S**tylesheet **L**anguage **T**ransformations*
- język przekształceń dla dokumentów XML
  - pozwala na przetłumaczenie dokumentów z jednego formatu XML na dowolny inny format zgodny ze składnią XML, np. na
    - stronę WWW XHTML
    - wzór matematyczny MathML
    - dokument biurowy ODF
    - zwykły HTML i czysty tekst
  - umożliwia wizualizację dokumentów XML
    - odpowiednik **CSS** (ang. **C**ascading **S**tyle **S**heets), kaskadowych arkuszy stylów dla dokumentów HTML
- standard opracowany przez W3C
- m.in. umożliwia
  - dodawanie/usuwanie elementów i atrybutów do/z dokumentu wynikowego
  - przestawianie i sortowanie elementów
  - ukrywanie i wyświetlanie elementów
- wykorzystuje **XPath** do lokalizacji elementów

JĘZYK XSLT

XSLT | PRZYKŁAD

dokument XML

```
<?xml version="1.0" encoding="UTF-8"?>
<Ewidencja_Jaskiń>
 <Jaskinia nr_inwentarzowy="G-7.27">
  <Nazwa>Rajska Brama</Nazwa>
  <Region>Region Swietokrzyski</Region>
  <Długość_m>4.5</Długość_m>
  <Głębokość_m>0.5</Głębokość_m>
 </Jaskinia>
 <Jaskinia nr_inwentarzowy="T.E-10.03">
  <Nazwa>Dziura nad Studnia</Nazwa>
  <Region>Tatry</Region>
  <Długość_m>20</Długość_m>
  <Głębokość_m>6.2</Głębokość_m>
 </Jaskinia>
 <Jaskinia nr_inwentarzowy="N-2.62">
  <Nazwa>Jaskinia Polkolista</Nazwa>
  <Region>Niecka Nidzianska</Region>
  <Długość_m>58</Długość_m>
  <Głębokość_m>1</Głębokość_m>
 </Jaskinia>
</Ewidencja_Jaskiń>
```

przekształcenie XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
  <html>
   <body>
    <h2>Jaskinie w Polsce</h2>
    <table border="1">
     <tr bgcolor="#FFFF00">
      <th>Nazwa Jaskini</th>
      <th>Region Polski</th>
     </tr>
     <xsl:for-each select="Ewidencja_Jaskiń/Jaskinia">
      <tr>
       <td><xsl:value-of select="Nazwa"/></td>
       <td><xsl:value-of select="Region"/></td>
      </tr>
     </xsl:for-each>
    </table>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

XSLT | PRZYKŁAD

przekształcony dokument XML

```
<html>
 <body>
  <h2>Jaskinie w Polsce</h2>
  <table border="1">
   <tr bgcolor="#FFFF00">
    <th>Nazwa Jaskini</th>
    <th>Region Polski</th>
   </tr>
   <tr>
    <td>Rajska Brama</td>
    <td>Region Swietokrzyski</td>
   </tr>
   <tr>
    <td>Dziura nad Studnia</td>
    <td>Tatry</td>
   </tr>
   <tr>
    <td>Jaskinia Polkolista</td>
    <td>Niecka Nidzianska</td>
   </tr>
  </table>
 </body>
</html>
```

Jaskinie w Polsce

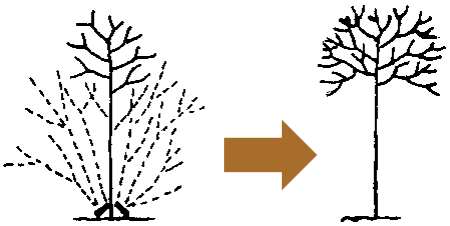
Nazwa Jaskini	Region Polski
Rajska Brama	Region Swietokrzyski
Dziura nad Studnia	Tatry
Jaskinia Polkolista	Niecka Nidzianska

XSL

- ang. *e**X**tensible **S**tylesheet **L**anguage*
- arkusze stylów dla dokumentów XML
  - określają sposób zamiany dokumentu XML na inny dokument XML, dla którego określony jest już sposób prezentacji
  - składają się z 4 części
    - *XSLT*
      - język przekształceń dla dokumentów XML
    - *XPath*
      - język do nawigacji w dokumentach XML
    - *XSL-FO*
      - język formatowania dokumentów XML (zaniechany w 2013 r.)
    - *XQuery*
      - język zapytań dla dokumentów XML

XSLT

- najważniejsza część XSL
- przekształcenie XSL
  - przekształca dokument XML w inny dokument XML
  - przekształca „drzewo” źródłowego dokumentu XML na „drzewo” docelowego dokumentu XML
  - wykorzystuje język XPath do odnajdywania elementów i atrybutów w dokumencie XML
  - wspierany przez większość przeglądarek internetowych



XSLT | ZASADA DZIAŁANIA

- XPath służy do zdefiniowania tych części dokumentu źródłowego, które powinny pasować do 1 lub kilku uprzednio zdefiniowanych szablonów
- kiedy dopasowania brak, przekształcona zostanie tylko część dokumentu źródłowego w dokument wynikowy

XSLT | DEKLARACJA

- element „korzeń” służący do deklaracji arkusza stylów (synonimy)
  - *<xsl:stylesheet>*

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

  - *<xsl:transform>*

```
<xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

- należy zadeklarować przestrzeń nazw XSLT (dostęp do elementów, atrybutów, funkcji XSLT)
  - *http://www.w3.org/1999/XSL/Transform*
    - oficjalna przestrzeń nazw W3C XSLT
    - dodatkowo wymagany atrybut wersja
      - *version="1.0"*

XML + XSLT | PRZYKŁAD

dokument XML

```
<?xml version="1.0" encoding="UTF-8"?>
<Ewidencja_Jaskiń>
 <Jaskinia nr_inwentarzowy="G-7.27">
  <Nazwa>Rajska Brama</Nazwa>
  <Region>Region Swietokrzyski</Region>
  <Długość_m>4.5</Długość_m>
  <Głębokość_m>0.5</Głębokość_m>
 </Jaskinia>
 <Jaskinia nr_inwentarzowy="T.E-10.03">
  <Nazwa>Dziura nad Studnia</Nazwa>
  <Region>Tatry</Region>
  <Długość_m>20</Długość_m>
  <Głębokość_m>6.2</Głębokość_m>
 </Jaskinia>
 <Jaskinia nr_inwentarzowy="N-2.62">
  <Nazwa>Jaskinia Polkolista</Nazwa>
  <Region>Niecka Nidzianska</Region>
  <Długość_m>58</Długość_m>
  <Głębokość_m>1</Głębokość_m>
 </Jaskinia>
</Ewidencja_Jaskiń>
```

arkusz stylów XSL (przekształcenie XSLT)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
  <html>
   <body>
    <h2>Jaskinie w Polsce</h2>
    <table border="1">
     <tr bgcolor="#FFFF00">
      <th>Nazwa Jaskini</th>
      <th>Region Polski</th>
     </tr>
     <xsl:for-each select="Ewidencja_Jaskiń/Jaskinia">
      <tr>
       <td><xsl:value-of select="Nazwa"/></td>
       <td><xsl:value-of select="Region"/></td>
      </tr>
     </xsl:for-each>
    </table>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

XML + XSLT | PRZYKŁAD

dokument XML + XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="Jaskinie_XSLT.xslt"?>
<Ewidencja_Jaskiń>
  <Jaskinia nr_inwentarzowy="G-7.27">
    <Nazwa>Rajska Brama</Nazwa>
    <Region>Region Swietokrzyski</Region>
    <Długość_m>4.5</Długość_m>
    <Głębokość_m>0.5</Głębokość_m>
  </Jaskinia>
  <Jaskinia nr_inwentarzowy="T.E-10.03">
    <Nazwa>Dziura nad Studnia</Nazwa>
    <Region>Tatry</Region>
    <Długość_m>20</Długość_m>
    <Głębokość_m>6.2</Głębokość_m>
  </Jaskinia>
   <Jaskinia nr_inwentarzowy="N-2.62">
    <Nazwa>Jaskinia Polkolista</Nazwa>
    <Region>Niecka Nidzianska</Region>
    <Długość_m>58</Długość_m>
    <Głębokość_m>1</Głębokość_m>
  </Jaskinia>
</Ewidencja_Jaskiń>
```


XSLT | ELEMENT <XSL:TEMPLATE>

- element <xsl:template>
  - służy do budowy szablonów (reguł)
  - zawiera reguły stosowane, gdy określony węzeł spełnia warunki dopasowania
    - arkusz stylów XSL składa się z 1 lub kilku szablonów
  - atrybut *match*
    - służy do skojarzenia szablonu z odpowiednim elementem XML
    - może służyć do zdefiniowania szablonu dla całego dokumentu XML
    - jego wartość jest wyrażeniem XPath
      - np. *match="/"*
      - określa cały dokument XML

XSLT | ELEMENT <XSL:TEMPLATE> | PRZYKŁAD

- *<?xml version="1.0" encoding="UTF-8"?>*
  - deklaracja XML (arkusz stylów XSL jest dokumentem XML)
- *<xsl:stylesheet>*
  - określa, że ten dokument jest dokumentem XSLT
    - wraz z wersją XSLT i przestrzenią nazw
- *<xsl:template>*
  - definiuje szablon
  - atrybut *match="/"*
    - dołącza szablon do „korzenia" źródłowego dokumentu XML
  - zawartość elementu <xsl:template> definiuje wygląd wynikowego dokumentu XML (tu: HTML)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
  <html>
   <body>
    <h2>Jaskinie w Polsce</h2>
    <table border="1">
     <tr bgcolor="#FFFF00">
      <th>Nazwa Jaskini</th>
      <th>Region Polski</th>
     </tr>
     <xsl:for-each select="Ewidencja_Jaskiń/Jaskinia">
      <tr>
       <td><xsl:value-of select="Nazwa"/></td>
       <td><xsl:value-of select="Region"/></td>
      </tr>
     </xsl:for-each>
    </table>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

JĘZYK XSLT

XSLT | ELEMENT <XSL:VALUE-OF>

- element <xsl:value-of> służy do
  - wydobywania wartości elementów XML
  - dodawania ich do pliku wynikowego

XSLT | ELEMENT <XSL:VALUE-OF> | PRZYKŁAD

- atrybut *select* zawiera wyrażenie XPath
  - działa jak nawigacja w systemie plików
    - znak ukośnika (/) umożliwia wybór podkatalogów


```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
  <html>
   <body>
    <h2>Jaskinie w Polsce</h2>
    <table border="1">
     <tr bgcolor="#FFFF00">
      <th>Nazwa Jaskini</th>
      <th>Region Polski</th>
     </tr>
     <xsl:for-each select="Ewidencja_Jaskiń/Jaskinia">
      <tr>
       <td><xsl:value-of select="Nazwa"/></td>
       <td><xsl:value-of select="Region"/></td>
      </tr>
     </xsl:for-each>
    </table>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

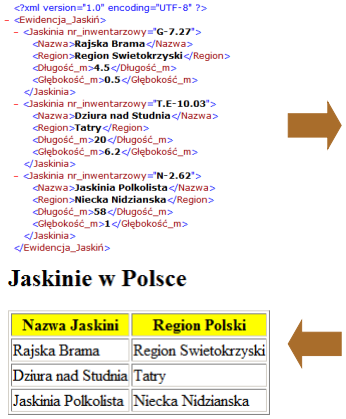

XSLT | ELEMENT <XSL:FOR-EACH>

- element <xsl:for-each> służy do
  - tworzenia pętli
  - wyboru wszystkich elementów XML w określonym zestawie węzłów
  - filtrowania wartości elementów XML
    - operatory filtrowania
      - = (równy)
      - != (różny od)
      - < (mniejszy niż)
      - > (większy niż)

    - np. *<xsl:for-each select="Ewidencja_Jaskiń/Jaskinia[Nazwa='Rajska Brama']">*

XSLT | ELEMENT <XSL:FOR-EACH> | PRZYKŁAD

- atrybut *select* zawiera wyrażenie XPath
  - działa jak nawigacja w systemie plików
    - znak ukośnika (/) umożliwia wybór podkatalogów




```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
  <html>
   <body>
    <h2>Jaskinie w Polsce</h2>
    <table border="1">
     <tr bgcolor="#FFFF00">
      <th>Nazwa Jaskini</th>
      <th>Region Polski</th>
     </tr>
     <xsl:for-each select="Ewidencja_Jaskiń/Jaskinia">
      <tr>
       <td><xsl:value-of select="Nazwa"/></td>
       <td><xsl:value-of select="Region"/></td>
      </tr>
     </xsl:for-each>
    </table>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```


XSLT | ELEMENT <XSL:FOR-EACH> | PRZYKŁAD

- filtrowanie wartości elementów XML

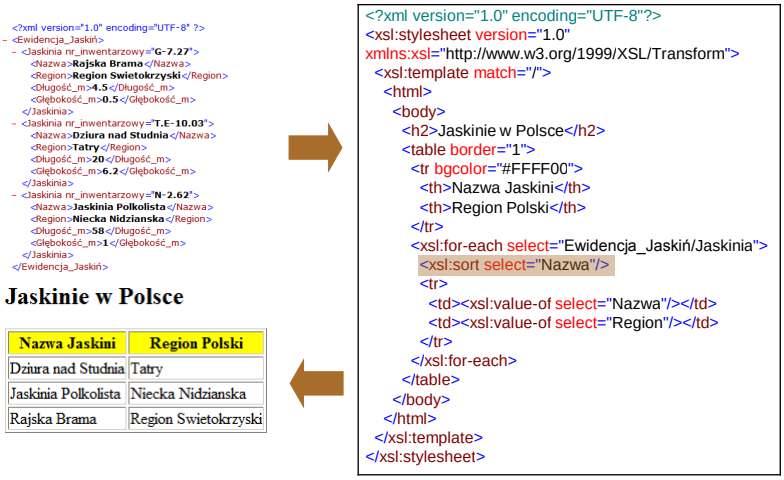

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
  <html>
   <body>
    <h2>Jaskinie w Polsce</h2>
    <table border="1">
     <tr bgcolor="#FFFF00">
      <th>Nazwa Jaskini</th>
      <th>Region Polski</th>
     </tr>
     <xsl:for-each select="Ewidencja_Jaskiń/Jaskinia[Nazwa='Rajska Brama']">
      <tr>
       <td><xsl:value-of select="Nazwa"/></td>
       <td><xsl:value-of select="Region"/></td>
      </tr>
     </xsl:for-each>
    </table>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

XSLT | ELEMENT <XSL:SORT>

- element <xsl:sort> służy do
  - sortowania elementów

XSLT | ELEMENT <XSL:SORT> | PRZYKŁAD

- atrybut *select* wskazuje, które elementy XML posortować



XSLT | ELEMENT <XSL:IF>

- element <xsl:if> służy do
  - definiowania i testowania warunków dla zawartości dokumentu XML

```
<xsl:if test="wyrażenie">
  ... wynik, jeśli wyrażenie jest prawdziwe ...
</xsl:if>
```


XSLT | ELEMENT <XSL:IF> | PRZYKŁAD

- atrybut *test* zawiera wyrażenie, które będzie testowane

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Ewidencja_Jaskiń>
  - <Jaskinia nr_inwentarzowy="G-7.27">
      <Nazwa>Rajska Brama</Nazwa>
      <Region>Region Swietokrzyski</Region>
      <Długość_m>4.5</Długość_m>
      <Głębokość_m>0.5</Głębokość_m>
    </Jaskinia>
  - <Jaskinia nr_inwentarzowy="T.E-10.03">
      <Nazwa>Dziura nad Studnia</Nazwa>
      <Region>Tatry</Region>
      <Długość_m>20</Długość_m>
      <Głębokość_m>6.2</Głębokość_m>
    </Jaskinia>
  - <Jaskinia nr_inwentarzowy="N-2.62">
      <Nazwa>Jaskinia Polkolista</Nazwa>
      <Region>Niecka Nidzianska</Region>
      <Długość_m>58</Długość_m>
      <Głębokość_m>1</Głębokość_m>
    </Jaskinia>
</Ewidencja_Jaskiń>
```

Jaskinie w Polsce

Nazwa Jaskini	Region Polski
Dziura nad Studnia	Tatry
Jaskinia Polkolista	Niecka Nidzianska

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
        <h2>Jaskinie w Polsce</h2>
        <table border="1">
          <tr bgcolor="#FFFF00">
            <th>Nazwa Jaskini</th>
            <th>Region Polski</th>
          </tr>
          <xsl:for-each select="Ewidencja_Jaskiń/Jaskinia">
            <xsl:if test="Długość_m &gt; 19">
              <tr>
                <td><xsl:value-of select="Nazwa"/></td>
                <td><xsl:value-of select="Region"/></td>
              </tr>
            </xsl:if>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

XSLT | ELEMENT <XSL:CHOOSE>

- element <xsl:choose> służy do
  - definiowania i testowania wielokrotnych warunków dla zawartości dokumentu XML
  - występuje w połączeniu z
    - elementem <xsl:when>
    - elementem <xsl:otherwise>

```
<xsl:choose>
  <xsl:when test="wyrażenie">
    ... jakiś wynik ...
  </xsl:when>
  <xsl:otherwise>
    ... jakiś wynik ...
  </xsl:otherwise>
</xsl:choose>
```


XSLT | ELEMENT <XSL:CHOOSE> | PRZYKŁAD

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
        <h2>Jaskinie w Polsce</h2>
        <table border="1">
          <tr bgcolor="#FFFF00">
            <th>Nazwa Jaskini</th><th>Region Polski</th>
          </tr>
          <xsl:for-each select="Ewidencja_Jaskiń/Jaskinia">
            <tr>
              <td><xsl:value-of select="Nazwa"/></td>
              <xsl:choose>
                <xsl:when test="Długość_m &gt; 19">
                  <td bgcolor="#EE82EE"><xsl:value-of select="Region"/></td>
                </xsl:when>
                <xsl:otherwise>
                  <td><xsl:value-of select="Region"/></td>
                </xsl:otherwise>
              </xsl:choose>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Ewidencja_Jaskiń>
  - <Jaskinia nr_inwentarzowy="G-7.27">
      <Nazwa>Rajska Brama</Nazwa>
      <Region>Region Swietokrzyski</Region>
      <Długość_m>4.5</Długość_m>
      <Głębokość_m>0.5</Głębokość_m>
    </Jaskinia>
  - <Jaskinia nr_inwentarzowy="T.E-10.03">
      <Nazwa>Dziura nad Studnia</Nazwa>
      <Region>Tatry</Region>
      <Długość_m>20</Długość_m>
      <Głębokość_m>6.2</Głębokość_m>
    </Jaskinia>
  - <Jaskinia nr_inwentarzowy="N-2.62">
      <Nazwa>Jaskinia Polkolista</Nazwa>
      <Region>Niecka Nidzianska</Region>
      <Długość_m>58</Długość_m>
      <Głębokość_m>1</Głębokość_m>
    </Jaskinia>
</Ewidencja_Jaskiń>
```

Jaskinie w Polsce

Nazwa Jaskini	Region Polski
Rajska Brama	Region Swietokrzyski
Dziura nad Studnia	Tatry
Jaskinia Polkolista	Niecka Nidzianska

XSLT | ELEMENT <XSL:CHOOSE> | PRZYKŁAD

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
  <html>
   <body>
    <h2>Jaskinie w Polsce</h2>
    <table border="1">
     <tr bgcolor="#FFFF00"><th>Nazwa Jaskini</th><th>Region Polski</th></tr>
     <xsl:for-each select="Ewidencja_Jaskiń/Jaskinia">
      <tr>
       <td><xsl:value-of select="Nazwa"/></td>
       <xsl:choose>
        <xsl:when test="Długość_m &gt; 50">
          <td bgcolor="#EE82EE">
           <xsl:value-of select="Region"/></td>
        </xsl:when>
        <xsl:when test="Długość_m &gt; 10">
          <td bgcolor="#C6CECE">
           <xsl:value-of select="Region"/></td>
        </xsl:when>
        <xsl:otherwise>
          <td><xsl:value-of select="Region"/></td>
        </xsl:otherwise>
       </xsl:choose>
      </tr>
     </xsl:for-each>
    </table>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Ewidencja_Jaskiń>
  - <Jaskinia nr_inwentarzowy="G-7.27">
      <Nazwa>Rajska Brama</Nazwa>
      <Region>Region Swietokrzyski</Region>
      <Długość_m>4.5</Długość_m>
      <Głębokość_m>0.5</Głębokość_m>
    </Jaskinia>
  - <Jaskinia nr_inwentarzowy="T.E-10.03">
      <Nazwa>Dziura nad Studnia</Nazwa>
      <Region>Tatry</Region>
      <Długość_m>20</Długość_m>
      <Głębokość_m>6.2</Głębokość_m>
    </Jaskinia>
  - <Jaskinia nr_inwentarzowy="N-2.62">
      <Nazwa>Jaskinia Polkolista</Nazwa>
      <Region>Niecka Nidzianska</Region>
      <Długość_m>58</Długość_m>
      <Głębokość_m>1</Głębokość_m>
    </Jaskinia>
</Ewidencja_Jaskiń>
```

Jaskinie w Polsce

Nazwa Jaskini	Region Polski
Rajska Brama	Region Swietokrzyski
Dziura nad Studnia	Tatry
Jaskinia Polkolista	Niecka Nidzianska

XSLT | ELEMENT <XSL:APPLY-TEMPLATES>

- element <xsl:apply-templatese> służy do
  - zastosowania szablonu do
    - węzła bieżącego elementu
    - węzła „dziecka” bieżącego elementu
  - atrybut *select*
    - przekształcenie tylko elementu „dziecko”, który odpowiada wartości tego atrybutu
    - określa kolejność przekształcania węzłów „dzieci”

XSLT | ELEMENT <XSL:APPLY-TEMPLATES> | PRZYKŁAD

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
  <html>
   <body>
    <h2>Jaskinie w Polsce</h2>
    <xsl:apply-templates/>
   </body>
  </html>
 </xsl:template>

 <xsl:template match="Jaskinia">
  <p>
   <xsl:apply-templates select="Nazwa"/>
   <xsl:apply-templates select="Region"/>
  </p>
 </xsl:template>
 <xsl:template match="Nazwa">Nazwa Jaskini:
          <span style="color:#C04000"><xsl:value-of select="."/></span><br/>
 </xsl:template>
 <xsl:template match="Region">Region Polski:
          <span style="color:#808000"><xsl:value-of select="."/></span><br/>
 </xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Ewidencja_Jaskiń>
  - <Jaskinia nr_inwentarzowy="G-7.27">
      <Nazwa>Rajska Brama</Nazwa>
      <Region>Region Swietokrzyski</Region>
      <Długość_m>4.5</Długość_m>
      <Głębokość_m>0.5</Głębokość_m>
    </Jaskinia>
  - <Jaskinia nr_inwentarzowy="T.E-10.03">
      <Nazwa>Dziura nad Studnia</Nazwa>
      <Region>Tatry</Region>
      <Długość_m>20</Długość_m>
      <Głębokość_m>6.2</Głębokość_m>
    </Jaskinia>
  - <Jaskinia nr_inwentarzowy="N-2.62">
      <Nazwa>Jaskinia Polkolista</Nazwa>
      <Region>Niecka Nidzianska</Region>
      <Długość_m>58</Długość_m>
      <Głębokość_m>1</Głębokość_m>
    </Jaskinia>
</Ewidencja_Jaskiń>
```

Jaskinie w Polsce

Nazwa Jaskini: Rajska Brama
Region Polski: Region Swietokrzyski

Nazwa Jaskini: Dziura nad Studnia
Region Polski: Tatry

Nazwa Jaskini: Jaskinia Polkolista
Region Polski: Niecka Nidzianska

JĘZYK XSLT

XSLT | ELEMENT <XSL:ATTRIBUTE>

- element <xsl:attribute> służy do
  - dodawania atrybutów do elementów

```
<Jaskinia>
  <xsl:attribute name="nr_inwentarzowy"/>
</Jaskinia>
```

XSLT | ELEMENT <XSL:COPY> & ELEMENT <XSL:COPY-OF>

- element <xsl:copy> służy do
  - tworzenia kopi bieżącego węzła
  - **!!!** „dzieci” i atrybuty bieżącego węzła **nie są** automatycznie kopiowane
- element <xsl:copy-of> służy do
  - tworzenia kopi bieżącego węzła
  - **!!!** „dzieci” i atrybuty bieżącego węzła **są** automatycznie kopiowane

```
<xsl:template match="/">
  <xsl:element name="Ewidencja_Jaskin">
    <xsl:for-each select="/Ewidencja_Jaskin/jaskinia">
      <xsl:copy-of select="document(@name)"/>
    </xsl:for-each>
  </xsl:element>
</xsl:template>
```

JĘZYK XSLT

XSLT | ELEMENT <XSL:TEXT>

- element <xsl:text> służy do
  - wprowadzania tekstu (dosłownego, literału) do pliku wynikowego

```
<xsl:for-each select="Ewidencja_Jaskin/Jaskinia">
  <xsl:value-of select="Nazwa"/>
  <xsl:if test="position() < last()-1">
    <xsl:text>, </xsl:text>
  </xsl:if>
  <xsl:if test="position() = last()-1">
    <xsl:text>, and </xsl:text>
  </xsl:if>
  <xsl:if test="position() = last()">
    <xsl:text>!</xsl:text>
  </xsl:if>
</xsl:for-each>
```

XSLT | ELEMENT <XSL:COMMENT>

- element <xsl:comment> służy do
  - tworzenia komentarzy w pliku wynikowym

```
<xsl:comment>Wersja testowa!</xsl:comment>
```

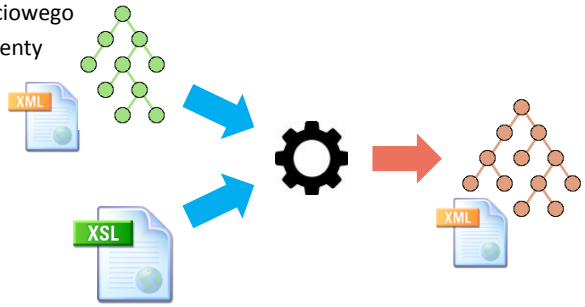

XSLT | QNAME

- *QName*
  - nazwa kwalifikowana, nazwa z przedrostkiem przestrzeni nazw
  - pełna nazwa kwalifikowana elementu, atrybutu, identyfikatora w dokumencie XML

```
<cbdg:Jaskinia xmlns:cbdg="http://cbdgmapa.pgi.gov.pl" nr_inwentarzowy="G-7.27"/>
```

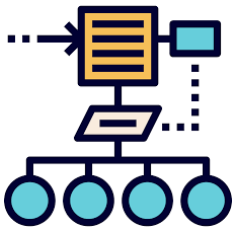
ALGORYTM TRANSFORMACJI PROCESORA XSLT

- **przygotowanie**
  - arkusz XSLT oraz wejściowy dokument XML są parsowane
    - analizowana jest ich struktura i budowane są drzewa dokumentów
  - z dokumentów usuwane są nadmiarowe białe znaki
  - do drzewa XSLT dołączane są standardowe reguły
- **transformacja**
  - tworzony jest główny element drzewa wyjściowego
  - przetwarzane są elementy drzewa wejściowego
  - zwracane jest drzewo wyjściowe



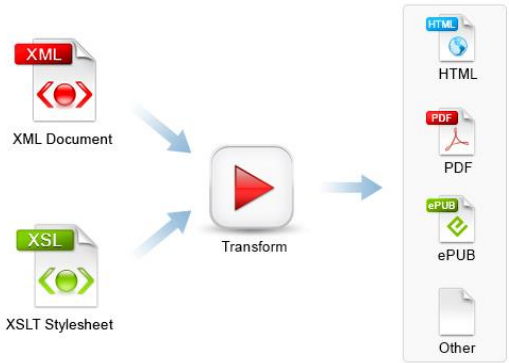
TRANSFORMACJA XSLT | PRZETWARZANIE ELEMENTÓW

- **przetwarzanie każdego elementu drzewa wejściowego**
  - znajdowany jest najlepiej pasujący szablon
  - znaleziony szablon jest stosowany
    - elementy szablonu znajdujące się w przestrzeni nazw XSLT (zwykle z prefiksem *xsl:*) traktowane są jak instrukcje i odpowiednio interpretowane
      - reszta elementów jest kopiowana do drzewa wynikowego
  - jeśli w szablonie
    - jest umieszczona instrukcja *xsl:apply-templates*
      - procesor przechodzi w tym miejscu do rekurencyjnego przetwarzania listy elementów wskazanych atrybutem *select* lub wszystkich dzieci aktualnego elementu (gdy brak atrybutu *select*)
    - brak jest instrukcji *xsl:apply-templates*
      - żadne z elementów aktualnego poddrzewa (dzieci i ich następniki) nie są w tym miejscu dopasowywane (przetwarzane)



DLACZEGO WARTO ZNAĆ XSLT?

- pozwala przekształcić dowolny **dokument XML** na **inny dokument XML**, np.
  - dokument XML na dokument XML zgodny z określonym schematem XSD
  - dokument XML na dokument GML



ĆWICZENIE 8: XSLT I

- **XSLT**
  (*apply*, *choose*, *for-each*, *if*, *sort*, *value-of*)
  - przeanalizować i przetestować przykładowe proste przekształcenia XSLT
    - *DANE_XML\XSLT\Przykłady\Proste*
  - wykorzystać aplikację *Notepad++*
    - wtyczka *XML Tools* → funkcja *XSL Transformation*
  - wykorzystać aplikację *XML Copy Editor*
    - *XML* → *XSL Transform…*
  - wykorzystać aplikację *AltovaXML Community Edition 2013*
    - *DANE_XML\XSLT\ProcesorAltova*

ĆWICZENIE 9: XSLT II

- **XSLT**
  (łączenie plików XML)
  - przeanalizować i przetestować przykładowe proste przekształcenia XSLT
    - *DANE_XML\XSLT\Przykłady\Proste\Merge*
  - wykorzystać aplikację *Notepad++*
    - wtyczka *XML Tools* → funkcja *XSL Transformation*
  - wykorzystać aplikację *XML Copy Editor*
    - *XML* → *XSL Transform…*
  - wykorzystać aplikację *AltovaXML Community Edition 2013*
    - *DANE_XML\XSLT\ProcesorAltova*

JĘZYK XSLT

ĆWICZENIE 10: XSLT III

- **XSLT**
  (*attribute*, *copy-of*, text, zmienne lokalne i globalne, pętle i warunki, funkcje wbudowane i własne)
  - przeanalizować i przetestować przykładowe złożone przekształcenia XSLT
    - *DANE_XML\XSLT\Przykłady\Złożone*
  - wykorzystać aplikację *Notepad++*
    - wtyczka *XML Tools* → funkcja *XSL Transformation*
  - wykorzystać aplikację *XML Copy Editor*
    - *XML* → *XSL Transform…*
  - wykorzystać aplikację *AltovaXML Community Edition 2013*
    - *DANE_XML\XSLT\ProcesorAltova*

ĆWICZENIE 11: XSLT III

- **XSLT**
  (opracowanie własnej transformacji)
  - pobrać rejestr pożarów z 2018 roku w formacie GML
    - rejestr pożarów dostępny na stronie *Instytutu Badawczego Leśnictwa*
      - *http://bazapozarow.ibles.waw.pl:8080/ibl-ppoz-web/export.xhtml*
  - dla pobranej próbki danych opracować arkusz stylów XSL, który przekształci plik z danymi na strukturę zgodną ze schematem aplikacyjnym dla tematu *Strefy zagrożenia naturalnego*
    - *DANE_XML\XSLT\Pożary\NaturalRiskZones\inspire_dataspecification_nz_v3.0.pdf*
  - ↳ ograniczyć próbkę danych źródłowych do kilku obiektów (np. 5)
  - ↳ ze schematu aplikacyjnego dla tematu *Strefy zagrożenia naturalnego* wystarczy odzwierciedlić tylko jedną klasę obiektów
    - np. *ObservedEvent*
  - ↳ na podstawie schematu aplikacyjnego GML (plik XSD) przygotować próbkę danych wynikowych (plik XML/GML)

DZIĘKUJĘ ZA UWAGĘ!!! ☺

JĘZYK SCHEMATRON

BONUS

JĘZYK SCHEMATRON

- język znaczników, oparty na regułach
- standard ISO
  - ISO/IEC 19757-3:2016 *Information technology – Document Schema Definition Languages (DSDL)* – Part 3: *Rule-based validation – Schematron*
- służy do walidacji dokumentów XML
  - umożliwia testowanie i sprawdzanie warunków, których nie obsługuje język XML Schema
  - nie jest konkurencją dla XML Schema
    - w Schematronie bardzo trudno i nieintuicyjnie zapisuje się reguły sprawdzające budowę strukturalną dokumentu
    - zaleca się, aby używać łącznie języka XML Schema i Schematron
      - w XML Schema można zamodelować strukturę dokumentu
      - w Schematronie reguły kontekstowe
- wykorzystuje **XPath** do lokalizacji elementów

SCHEMATRON | BLOKI KONSTRUKCYJNE

- **asercje (*assertions*)**
  - opisują testowane warunki
- **komunikaty (*messages*)**
  - informują o zdanym/niezdanym teście
- **reguły (*rules*)**
  - zbiór testów, które są mają zastosowanie do określonych elementów XML (kontekst – *context*)
- **wzorce (*patterns*)**
  - pogrupowane reguły
- **fazy (p*hases*)**
  - aktywują różne wzorce w określonym czasie

JĘZYK SCHEMATRON

SCHEMATRON | PRZYKŁAD

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
  <title>Test for dog</title>
  <pattern id="example">
    <rule context="dog">
      <assert test="bone">Give that dog a bone!</assert>
      <report test="flea">Your dog has fleas!</report>
    </rule>
  </pattern>
</schema>
```

SCHEMATRON | STRUKTURA

```
<schema>
  <title>
  <pattern> +
    <rule> +
      <assert> or <report> +
```

schema	korzeń dokumentu XML, zawiera pozostałe elementy
title	opisowy tytuł (dla użytkownika)
pattern	zbiór powiązanych reguł
rule	jedna lub więcej asercji mających zastosowanie w danym kontekście
assert, report	testy – deklarują warunki, które będą testowane (w ich atrybutach) i dostarczają komunikaty, które będą wyświetlane (w ich kontekście)

JĘZYK SCHEMATRON

SCHEMATRON | PRZYKŁAD (XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<dog>
  <flea/>
  <flea/>
  <bone/>
</dog>
```

dokument XML

SCHEMATRON | PRZYKŁAD (XSD)

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified">
 <element name="dog">
  <complexType>
   <sequence>
    <element name="flea" maxOccurs="unbounded"/>
    <element name="bone" minOccurs="0"/>
   </sequence>
  </complexType>
 </element>
</schema>
```

dokument XSD

JĘZYK SCHEMATRON

SCHEMATRON | PRZYKŁAD (SCH)

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
  <title>Test for dog</title>
  <pattern id="example">
    <rule context="dog">
      <assert test="bone">Give that dog a bone!</assert>
      <report test="flea">Your dog has fleas!</report>
    </rule>
  </pattern>
</schema>
```

dokument SCH

SCHEMATRON | PRZYKŁAD (WALIDACJA)

```
<?xml version="1.0" encoding="UTF-8"?>
<dog>
   <flea/>
   <flea/>
   <bone/>
</dog>
```

Your dog has fleas!

```
<?xml version="1.0" encoding="UTF-8"?>
<dog>
   <flea/>
   <flea/>
   <!--<bone/>-->
</dog>
```

Validation:
- Give that dog a bone!
- Your dog has fleas!

JĘZYK SCHEMATRON

SCHEMATRON | INTERPRETACJA

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
   <title>Test for dog</title>
   <pattern id="example">
      <rule context="dog">
         <assert test="bone">Give that dog a bone!</assert>
         <report test="flea">Your dog has fleas!</report>
      </rule>
   </pattern>
</schema>
```

- wzorzec (*pattern*) z 1 regułą (*rule*)
- reguła (*rule*) zawiera 2 testy (*tests*)
- reguła ma zastosowanie do elementów *dog* (kontekst – *context*)
  - każdy element *dog* musi mieć co najmniej 1 element *bone*
    - w kontekście *dog*, element *bone* musi być obecny
    - w przeciwnym wypadku asercja (assertion) nie zda testu i pojawi się komunikat (*message*)
  - każdy element dog może mieć element *flea*
    - w kontekście *dog*, jeżeli pojawi się element *flea*, zostanie wyświetlony raport

ĆWICZENIE 12: SCHEMATRON

- **Schematron**
  - przeanalizować przykładowe reguły walidacji dokumentów XML, zapisane w języku Schematron
    - *DANE_XML\Schematron*
  - wykorzystać dowolny edytor plików XML