

# Hipertekst i Hipermedia

## Projekt

**Temat:**

**MOJE HOBBY**

**Etapy:**

<b>Etap</b>	<b>Punktacja [pkt]</b>
HTML, Dokument XML, XML Schema	20
XSLT + FO	20

**Etap 1:** HTML (8pkt), Dokument XML, XML Schema (12pkt)

**HTML: (8 pkt)**

*Wymagania:*

- zawartość strony zgodna z tematem projektu
- HTML5
- walidowanie strony **(0,8pkt)**
- układ strony:
  - podział strony na kilka elementów (nagłówek, menu, stopka, pole z treścią) **(0,8pkt)**
- rozdzielenie treści na kilka plików (przynajmniej trzy) **(0,8pkt)**
- menu zawierające przynajmniej trzy opcje, a jedna z nich z dodatkowymi opcjami podrzędnymi; zaznaczanie wybranej opcji **(0,4pkt)**
- umieszczenie na stronie multimediów:
  - grafika
    - galeria zdjęć (bitmapy) (przynajmniej 5) ma być zorganizowana w postaci miniatur, które można obejrzeć powiększone **(0,6pkt)**
    - grafika wektorowa SVG (grafika może być z internetu, samemu należy dodać jej transformację i prostą animację) **(0,6 pkt)**
    - animacja (wykorzystanie mechanizmów HTML5, CSS) **(0,4pkt)**
- umieszczenie na stronie:
  - tabeli **(0,4pkt)**
  - adresu e-mail z możliwością wysłania poczty **(0,4pkt)**
  - odsyłaczy do innych stron internetowych **(0,2pkt)**
  - odsyłacza do wybranego miejsca w tekście lub do początku strony (wyświetlony tekst powinien być odpowiednio długi, aby była możliwość zademonstrowania tej opcji) **(0,2pkt)**
- style należy zdefiniować w oddzielnym arkuszu stylów, wykorzystać mechanizm CSS
  - różne style dla przynajmniej 4 selektorów (grup selektorów) **(0,6 pkt)**
  - klasy (przynajmniej 3) **(0,8pkt)**
  - identyfikator (przynajmniej 1) **(0,2pkt)**
- stworzenie prostej ankiety-formularza **(0,8pkt)**
  - przynajmniej 7 pól do wprowadzania danych
  - przynajmniej 4 różne rodzaje pól umożliwiających wprowadzanie danych,
  - przyciski do czyszczenia zawartości formularza oraz wysyłania danych
- dbałość o estetyczny wygląd strony

**XML, XML SCHEMA: (12pkt)**

*Wymagania:*

- Utworzyć plik w formacie XML zawierający dane związane z tematem projektu. W pliku muszą znaleźć się zdjęcia oraz linki.

- Dla pliku XML, aby wymusić jego odpowiednią składnię, należy zaprojektować i utworzyć plik XML Schema.
- Plik XML musi być poprawny składniowo i semantycznie. Struktura pliku XML musi być zgodna z podaną w XML Schema. Do sprawdzenia poprawności należy użyć walidatora (<http://tools.decisionsoft.com/schemaValidate/>).
- Dla stworzonego pliku XML wygenerować XML Schema przy użyciu Visual C++. Na zaliczenie projektu należy przynieść zarówno XML Schema stworzony przez siebie, jak i wygenerowany automatycznie.
- Należy również zwrócić uwagę na postać dokumentu, czyli sposób zapisu, stosowanie wcięć obrazujących strukturę danych, odpowiednie (adekwatne do zawartej w nich treści) nazywanie znaczników, atrybutów.

### **Wymagania szczegółowe:**

W pliku XML Schema należy zadeklarować i wykorzystać:

- co najmniej 6 definicji globalnych typów złożonych **(1,6pkt)**
- przynajmniej 5 definicji globalnych typów prostych **(1,6pkt)**
- co najmniej 2 definicje lokalnych typów złożonych **(0,8pkt)**
- przynajmniej 2 definicje lokalnych typów prostych **(0,8pkt)**
- stosowanie różnych modeli wyboru, mieszanego typu zawartości **(0,4pkt)**
- przynajmniej jedna definicja grupy (elementów lub atrybutów) **(0,4pkt)**
- istnienie przynajmniej 4 poziomów zagłębienia w strukturze dokumentu **(0,4pkt)**
- definicja przynajmniej 5 atrybutów z czego przynajmniej 1 zdefiniowany globalnie i użyty przynajmniej 2 razy **(1,2pkt)**
- różnorodne definicje przynajmniej 10 różnych elementów **(1,6pkt)**
- stosowanie aspektów (ograniczeń na elementy i atrybuty)
  - *length, minLength, maxLength, maxInclusive, minInclusive, maxExclusive, minExclusive*, (wybrane min 4) **(0,4pkt)**
  - *pattern, enumeration* **(0,8pkt)**
- wyprowadzanie typów **(0,4pkt)**
  - *extension* (rozszerzenie o dodatkowe elementy)
- przynajmniej 3 odnośniki do elementów i/lub atrybutów (ma być odniesienie i do atrybutu i do elementu) **(0,8pkt)**
- użycie listy **(0,4pkt)**
- wykorzystanie kombinacji (*union*) **(0,4pkt)**
- walidowanie pliku
- w pliku XML przynajmniej 3 wypełnione podelementy korzenia

### **Wybrane przykładowe błędy występujące w schematach:**

- błędy walidacji (plik się nie waliduje) **(do -10pkt)**
- trywialna definicja typu prostego (np. typ prosty, który jest zwykłym typem string) **(-2pkt)**
- powtarzanie definicji typów (wielokrotne definiowanie typów) **(-2pkt)**
- wykorzystanie anyType **(do -10pkt)**
- nieznaczenie przerobiony, wygenerowany plik xsd **(do -10pkt)**
- niepoprawne definiowanie elementów, atrybutów, struktury **(do -6pkt)**
  - np. zamiast używać maxOccurs=4, czterokrotne definiowanie takiego samego elementu
- brak zdjęć (w XML (min 4) oraz w Schema) **(-1pkt)**
- brak linków (w XML (min 4) oraz w Schema) **(-1pkt)**
- brak w pliku XML przynajmniej 3 wypełnionych podelementów korzenia **(-2pkt)**

### **Uwaga**

- Ostateczna liczba punktów za projekt jest uzależniona od odpowiedzi udzielanych podczas oddawania projektu, orientacji w projekcie i obowiązujących zagadnieniach.
- Podczas oddawania projektu **kod** ma być **pozbawiony** wszelkich **komentarzy**

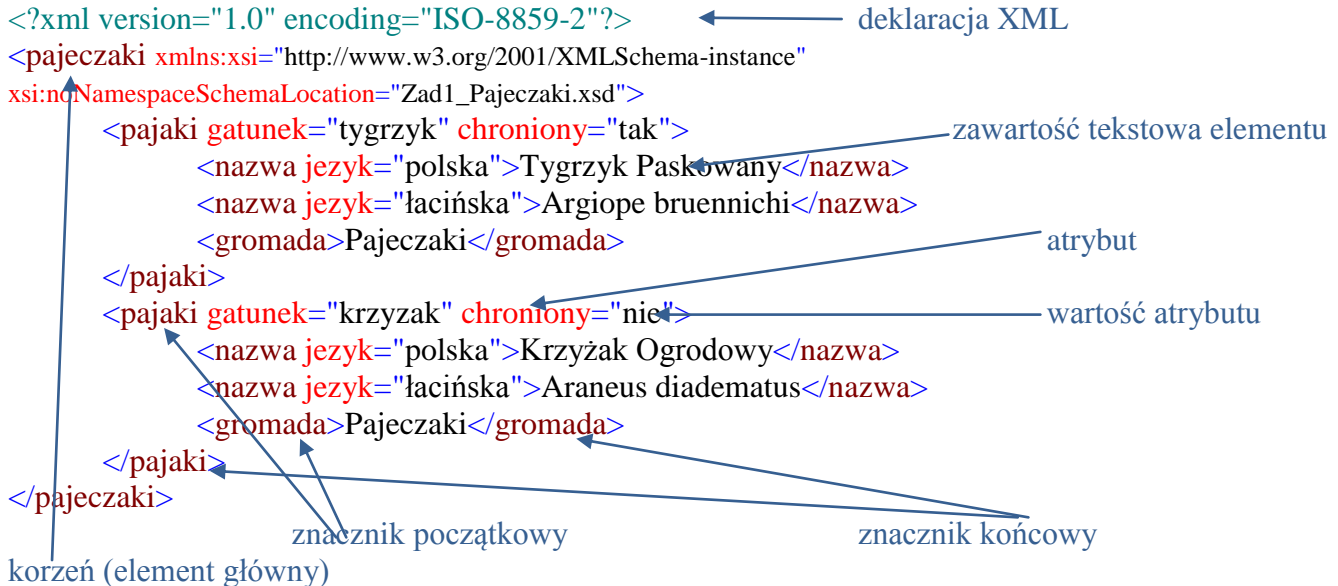
### **Oddawanie projektów**

- każda osoba ma wyznaczony na odbiór projektu termin: dzień, godzinę, salę oraz prowadzącego. Odbiór projektu odbywa się tylko w wyznaczonym terminie.
- należy przyjść 10 minut przed terminem oddawania projektu, aby wgrać pliki na komputer, otworzyć stronę walidator, itp.
- w czasie oddawania projektu należy otworzyć w Visual Studio pliki: html, css, XML, Schema własne oraz Schema wygenerowane
- nie ma możliwości poprawiania oddanych projektów

- XML i XML Schema - krótka ściągą ☺

## XML

- wszystkie niepuste elementy muszą mieć znacznik początkowy i końcowy
- elementy mogą być zagnieżdżone, nie mogą na siebie zachodzić
- rozróżnianie dużych i małych liter



## XML Schema

Jeśli chcemy stworzyć:

- tylko element z zawartością tekstową
  - typ prosty
- element z podelementami
  - typ złożony
- element z podelementami i atrybutami
  - typ złożony
- element z zawartością mieszaną (podelementy i tekst)
  - typ złożony z atrybutem `mixed=true`
- element z atrybutami
  - typ złożony
- element z atrybutami i zawartością tekstową
  - `simpleContent`

### 1) Definicja typu prostego nazwanego

```

<xs:simpleType name="krotki_string">
  <xs:restriction base="string">
    <xs:maxLength value="20" />
  </xs:restriction>
</xs:simpleType>
  
```

Annotations with arrows pointing to the corresponding parts of the XML:

- typ prosty nazwany**: points to the `<xs:simpleType name="krotki_string">` declaration.
- ograniczenie**: points to the `<xs:restriction base="string">` element.

## 2) Definicja elementu

Diagram illustrating the structure of an XML element definition with annotations:

- `<xs:element name="pajaki" maxOccurs="unbounded">`: **liczba wystąpień** (number of occurrences) points to `maxOccurs="unbounded"`; **definicja elementu** (element definition) points to the opening tag.
- `<xs:complexType>`: **typ złożony, lokalny** (complex type, local) points to the tag.
- `<xs:sequence>`: **sekwencja, elementy w ściśle określonej kolejności** (sequence, elements in strictly defined order) points to the tag.
- `<xs:element name="nazwa" maxOccurs="unbounded">`: **definicja elementu** (element definition) points to the opening tag.
- `<xs:complexType>`: **typ złożony, lokalny** (complex type, local) points to the tag.
- `<xs:attribute name="jezyk" type="xs:string" />`: **typ atrybutu** (attribute type) points to `type="xs:string"`; **definicja atrybutu (zawsze po definicjach elementów)** (attribute definition (always after element definitions)) points to the entire attribute tag.
- `</xs:complexType>`: **typ złożony, lokalny** (complex type, local) points to the closing tag.
- `</xs:element>`: **definicja elementu** (element definition) points to the closing tag.

## 3) Wyliczenia - lista predefiniowanych wartości

```
<xs:simpleType name="nazwa_typu" >  
  <xs:restriction base="string">  
    <xs:enumeration value="wartosc1" />  
    <xs:enumeration value="wartosc2" />  
    <xs:enumeration value="wartosc3" />  
  </xs:restriction>  
</xs:simpleType>
```

## 4) SimpleContent

Gdy stworzymy pochodny typ złożony na podstawie typu prostego lub innego typu złożonego o zawartości prostej. Można w ten sposób np. dodać atrybuty do typu bazowego.

```
<xs:complexType name="nazwa_typu">  
  <xs:simpleContent>  
    <xs:extension base="xs:string">  
      <xs:attribute name="nazwa_atrybutu" type="xs:string"/>  
    </xs:extension>  
  </xs:simpleContent>  
</xs:complexType>
```

## 5) Odniesienia do elementu

```
<xs:element name="data" type="xs:date"/>
```

globalna definicja elementu

```
<xs:element ref="data" minOccurs="0"/>
```

odniesienie do elementu zdefiniowanego globalnie