Stefan Popiołek
Projekt
Temat:Sklep z grami
Repozytorium:https://github.com/StudentStefan/ProjektXML
Nie za bardzo wiem co miałbym o tym powiedzieć
więc tylko pokażę że się waliduje z zewnętrznym dtd

```
~/Pulpit/xml/PROJEKT/ProjektXML$ xmllint --noout --valid SklepZgrami.xml
~/Pulpit/xml/PROJEKT/ProjektXML$
```

Przy czym jako że używamy opcji ID i IDREF jeśli zmienimy atrybut id w produkcie na jakiś nie istniejący to plik się nie zwaliduje

```
stefan@stefan-Inspiron-5577:~/Pulpit/xml/PROJEKT/ProjektXML$ xmllint --noout --valid SklepZgrami.xml
SklepZgrami.xml:153: element produkt: validity error : IDREF attribute id references an unknown ID "ara02"
stefan@stefan-Inspiron-5577:~/Pulpit/xml/PROJEKT/ProjektXML$
```

innymi słowy ID i IDREF działa prawidłowo
ID powoduje też że wartości atrybutu muszą być unikatowe dlatego gdy w pliku zmienimy wartość atrybutu id w grze drugiej od góry na wartość id pierwszego elementu też plik się nie zwaliduje
Wyskoczą 2 błędy jeden dotyczący powtarzającej się wartości która powinna być unikatowa a drugi dotyczący nieprawidłowego odwołania za pomocą IDREF w zamówieniach .

```
stefan@stefan-Inspiron-5577:~/Pulpit/xml/PROJEKT/ProjektXML$ xmllint --noout --valid SklepZgrami.xml
SklepZgrami.xml:13: element gra: validity error : ID gra01 already defined
                        <gra id="gra01">
                                        ^
SklepZgrami.xml:153: element produkt: validity error : IDREF attribute id references an unknown ID "gra02"
stefan@stefan-Inspiron-5577:~/Pulpit/xml/PROJEKT/ProjektXML$
```

Jako iż chciałem by ceny mogły być podawane w różnych walutach dałem cenie atrybut waluta który nie musi być podany bo przez dtd nada walutę domyślną w zł

```
<!ATTLIST cena waluta (zl|USD|euro) "zl">
```

Więc nawet jeśli usuniemy atrybut waluta to plik się zwaliduje aczkolwiek jeśli nadamy wartość inną niż zl USD czy euro to podczas walidacji otrzymamy błąd.

```
stefan@stefan-Inspiron-5577:~/Pulpit/xml/PROJEKT/ProjektXML$ xmllint --noout --valid SklepZgrami.xml
SklepZgrami.xml:17: element cena: validity error : Value "z" for attribute waluta of cena is not among the enumerated set
                                <cena waluta="z">179.99</cena>
                                               ^
stefan@stefan-Inspiron-5577:~/Pulpit/xml/PROJEKT/ProjektXML$
```

Jeśli chodzi o DTD to to już chyba wszystkie ciekawsze rzeczy

Przejdźmy do Schemy

No to najpierw udowodnię że xml się prawidłowo waliduje z xsd

```
stefan@stefan-Inspiron-5577:~/Pulpit/xml/PROJEKT/ProjektXML$ xmllint --noout --schema SklepZgrami.xsd SklepZgrami.xml
SklepZgrami.xml validates
stefan@stefan-Inspiron-5577:~/Pulpit/xml/PROJEKT/ProjektXML$
```

Są tutaj na dobrą sprawę dwie rzeczy godne uwagi :klucz i referencje do klucza oraz jednoczesna „restrykcja” i hmm „rozbudowa elementu”
Ale zacznijmy od kluczy dowód że działają zobaczyć można co prawda dopiero przy transformacji

```
<xsd:element name="sklep_z_grami" type="sklep_typ">
        <xsd:key name="kluczyk">
                <xsd:selector xpath="produkty/akcesoria/mysz |  produkty/akcesoria/klawiatura | produkty/akcesoria/sluchawki | produkty/gry/gra"/>
                <xsd:field xpath="@id"/>
        </xsd:key>
        <xsd:keyref name="kluczyk_ref" refer="kluczyk">
                <xsd:selector xpath="sklep_z_grami/zamówienia/zamówienie/produkt"/>
                <xsd:field xpath="@id"/>
        </xsd:keyref>
</xsd:element>
```

no ale jest to ważny element kodu .

Drugim może nie ważnym ale hmm trudniejszym? Jest element cena który ma jednocześnie restrykcje do typu decimal jak i dwóch możliwych cyfr po przecinku jak i atrybut waluta

```
<xsd:complexType name="cena_typ">
        <xsd:simpleContent>
                <xsd:extension base="cena_pomoc_typ">
                        <xsd:attribute name="waluta" type="waluta_typ"/>
                </xsd:extension>
        </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="cena_pomoc_typ">
<xsd:restriction base="xsd:decimal">
                <xsd:fractionDigits value="2"/>
        </xsd:restriction>
</xsd:simpleType>
```

Warto też pewnie wspomnieć ,że aby plik xsd był choć odrobinę bardziej czytelny (aczkolwiek jest to naprawdę mała różnica) podzieliłem go na typy .
Oraz żeby nie było że nie mam :
-restrykcja (zdjęcie wyżej)
-indikators (sequence,min/maxOccurs...)

```
<xsd:sequence>
        <xsd:element name="produkt" type="produkt_typ" minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element name="dane_osobowe" type="dane_osobowe_typ" minOccurs="1" maxOccurs="1"/>
</xsd:sequence>
<xsd:attribute name="id_zam" type="id_zam_typ" use="required"/>
```

-wzorce

```
<xsd:simpleType name="kod_pocztowy_typ">
        <xsd:restriction base="xsd:string">
                <xsd:pattern value="[0-9]{2}[-][0-9]{3}"/>
        </xsd:restriction>
</xsd:simpleType>
```

To teraz może XSLT
Tradycyjnie najpierw dowód , że działa (przy użyciu strony https://www.freeformatter.com/xsl-transformer.html) . Cały output w SkelpZgrami.html

Transformed XML:

```
<html>
   <link rel="stylesheet" type="text/css" href="SklepZgrami.css"/>
   <head>
      <title>Sklep Z Grami</title>
   </head>
   <body>
      <h2>Sklep Z Grami</h2>
      <div class="produkty">
         <h1>Produkty</h1>
         <div class="gry">
            <h2>Gry</h2>
            <div class="Filtry">
               <h4>Filtry</h4>
      Nazwa: <input type="text"
                    id="szukaj_gry_nazwa"
                    onkeyup="szukaj_Gry()"
                    placeholder="Podaj nazwę"
                    title="szukaj w grach"/>
      Gatunek: <input type="text"
```

Wydaje mi się że warto przy okazji tego wspomnieć o opcji mode
Dzięki której ten sam element może być przetwarzany na wiele różnych sposobów

```
<xsl:value-of select="./nazwa"/></td>
<xsl:value-of select="./firma"/></td>
<xsl:apply-templates select="./cena" mode="zl"/></td>
<xsl:apply-templates select="./cena" mode="USD"/></td>
<xsl:apply-templates select="./cena" mode="euro"/></td>
```

Dopiero w xslt można zobaczyć czy klucze działają prawidłowo odwołanie do nich wygląda mniej więcej tak

```
<xsl:apply-templates select="key('klucz_do_produktu',@id)/cena" mode="zl"/>
```

reszta raczej standardowa
xsl:if jest format number w tym wypadku zaokrągla do 2 cyfr po przecinku

```
<xsl:if test="@waluta='zl'">
        <xsl:value-of select="."/>
</xsl:if>
<xsl:if test="@waluta='USD'">
        <xsl:value-of select='format-number(.*4.14,"#.00")'/>
</xsl:if>
<xsl:if test="@waluta='euro'">
        <xsl:value-of select='format-number(.*4.52,"#.00")'/>
</xsl:if>
```

korzystam z xsl:for-each aby łatwo i ładnie zrobić tabelkę

```
    <tr>
            <th>Nazwa</th>
            <th>Typ</th>
            <th>Długość kabla</th>
            <th>Firma</th>
            <th>Cena zl</th>
            <th>Cena USD</th>
            <th>Cena euro</th>
    </tr>
    <xsl:for-each select="./klawiatura">
            <tr>
                    <td><xsl:value-of select="./nazwa"/></td>
                    <td><xsl:value-of select="./typ"/></td>
                    <td><xsl:value-of select="concat(./kabel,./kabel/@miara)"/></td>
                    <td><xsl:value-of select="./firma"/></td>
                    <td><xsl:apply-templates select="./cena" mode="zl"/></td>
                    <td><xsl:apply-templates select="./cena" mode="USD"/></td>
                    <td><xsl:apply-templates select="./cena" mode="euro"/></td>
            </tr>
    </xsl:for-each>
</table>
```

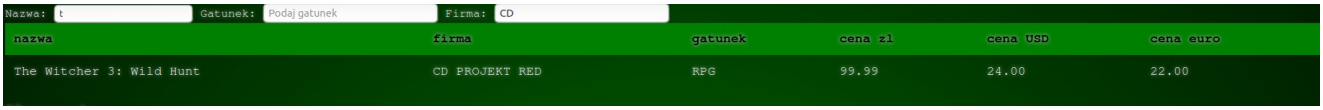
Mam pola i przyciski które uruchamiają skrypty zapisane w pliku SklepZgrami.js
warto też napomknąć że przy okazji pola input weryfikują choć trochę wprowadzane dane

```
 Nazwa: <input type="text" id="nazwa_sluchawki" placeholder="Podaj nazwę" title="Nadaj nazwę"/>
Firma: <input type="text" id="firma_sluchawki" placeholder="Podaj firmę" title="Nadaj firmę"/>
Cena zl: <input type="number" id="cena_sluchawki" placeholder="Podaj cenę" title="Nadaj cenę" min="0.22" step="0.01" oninput="validity.valid||(value='');"/>
<button onclick="dodajSluchawki()">Dodaj</button>
       Nr Kolumny <input type="number" id="rzad_sluchawki" placeholder="Podaj rząd" title="Zmień rząd" min="1" step="1"/>
       <button id="update_button" onclick="update()">Update</button>
```

A skoro już przy js jesteśmy
to pora na funkcje Przed Sortowaniem

Po wpisaniu do pola firma ‘CD’

I po wpisaniu literki ‘t’ do pola nazwa



Innymi słowy przeszukiwanie działa sortując jednocześnie po tych trzech polach
Wyniki które zostały odrzucone przez filtry są niejako chowane , filtr tez nie bierze pod uwagę wielkości znaków
A to implementacja tego filtra

```
function szukaj_Gry() {

  var input0,input1,input2,filter0,filter1,filter2,table,tr,td,i,txtValue0,txtValue1,txtValue2;
  input0 = document.getElementById("szukaj_gry_nazwa");
  input1 = document.getElementById("szukaj_gry_firma");
  input2 = document.getElementById("szukaj_gry_gatunek");
  filter0 = input0.value.toUpperCase();
  filter1 = input1.value.toUpperCase();
  filter2 = input2.value.toUpperCase();
  table = document.getElementById("gry_tab");
  tr = table.getElementsByTagName("tr");

  for (i = 0; i < tr.length; i++) {
    td0 = tr[i].getElementsByTagName("td")[0];
    td1 = tr[i].getElementsByTagName("td")[1];
    td2 = tr[i].getElementsByTagName("td")[2];
    if (td0) {
      txtValue0 = td0.textContent || td0.innerText;
      txtValue1 = td1.textContent || td1.innerText;
      txtValue2 = td2.textContent || td2.innerText;
      if (txtValue0.toUpperCase().indexOf(filter0) > -1 && txtValue1.toUpperCase().indexOf(filter1) > -1 && txtValue2.toUpperCase().indexOf(filter2) > -1 ) {
        tr[i].style.display = "";
      } else {
        tr[i].style.display = "none";
      }
    }

  }
}
```

No to tyle jeśli chodzi o filtry
czas na Dodawanie
żeby dodać słuchawki trzeba wypełnić wszystkie istotne pola jak nazwa firma i cena bo bez tego ani rusz dlatego gdy ktoś zapomni coś wpisać funkcja po uruchomieniu poinformuje użytkownika , ze zapomniał o jakimś polu

ale jeśli wszystko się uzupełni to funkcja ładnie wstawi wynik do tablicy , warto dodać że funkcja przy okazji oblicza ile dany produkt będzie kosztował w USD i euro

a oto implementacja

```
function dodajSłuchawki(){
        var nazwa,firma,cena,tabela,rzad,kolmuna_nazwa,kolumna_firma,kolumna_cena_zl,kolumna_cena_USD,kolumna_cena_euro;
        nazwa=document.getElementById("nazwa_sluchawki").value;
        firma=document.getElementById("firma_sluchawki").value;
        cena=document.getElementById("cena_sluchawki").value;
        if( nazwa==="" ){
                alert("Podaj nazwę");
        }else if(firma===""){
                alert("Podaj firmę")
        }else{
                tabela=document.getElementById("sluchawki_tab");
                rzad=tabela.insertRow();
                kolumna_nazwa=rzad.insertCell(0);
                kolumna_firma=rzad.insertCell(1);
                kolumna_cena_zl=rzad.insertCell(2);
                kolumna_cena_USD=rzad.insertCell(3);
                kolumna_cena_euro=rzad.insertCell(4);
                kolumna_nazwa.innerHTML=nazwa;
                kolumna_firma.innerHTML=firma;
                kolumna_cena_zl.innerHTML=cena;
                kolumna_cena_USD.innerHTML=(cena*0.24).toFixed(2);
                kolumna_cena_euro.innerHTML=(cena*0.22).toFixed(2);
        }
}
```

Przy okazji dodawania postanowiłem wykorzystać już pola input (tak wiem jestem leniem) do Update'owania , wystarczy podać nr rzędu i wpisać nową wartość w pola które chce się zmienić . Tutaj muszę wspomnieć o tym iż internet mnie okłamał bo twierdził że nie da się zmodyfikować komórki w tabeli …. Jednak się da i to całkiem prosto … (stąd jak Pan zobaczy commity to mam o sporo usuniętych linijek w js…)

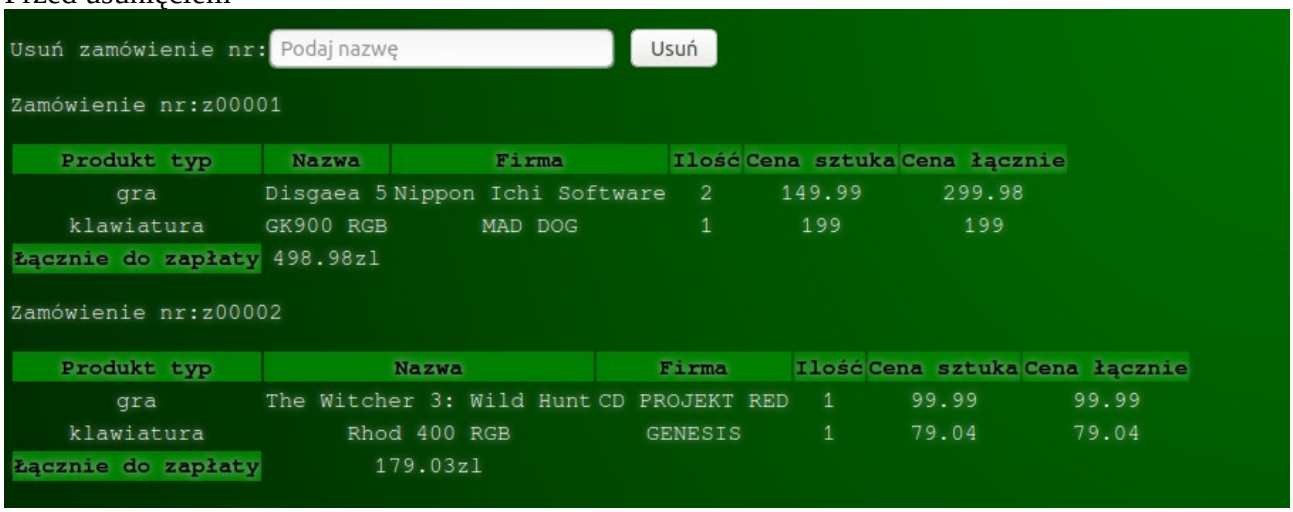
a to implementacja

```
function update(){
        var nazwa,firma,cena,tabela;
        nazwa=document.getElementById("nazwa_sluchawki").value;
        firma=document.getElementById("firma_sluchawki").value;
        cena=document.getElementById("cena_sluchawki").value;
        tabela=document.getElementById("sluchawki_tab");
        var nr=document.getElementById("rzad_sluchawki").value;
        var rzedy=tabela.getElementsByTagName("tr");
        if (tabela.rows.length-1<nr || nr<=0){
                alert("Nie ma takiego rzędu");
        }else{
                if(nazwa.length!=0){
                        rzedy[nr].cells[0].innerHTML=nazwa;
                }
                if(firma.length!=0){
                        rzedy[nr].cells[1].innerHTML=firma;
                }
                if(cena.length!=0){
                        rzedy[nr].cells[2].innerHTML=cena;
                        rzedy[nr].cells[3].innerHTML=(cena*0.24).toFixed(2);
                        rzedy[nr].cells[4].innerHTML=(cena*0.22).toFixed(2);
                }

        }
}
```

No i ostatnie najłatwiejsze, usuwanie tutaj jedyne co mnie trochę zaskoczyło to to że trzeba przejść do rodzica żeby usunąć jakiś element (ma to sens ale hmm nie spodziewałem się tego) i zabezpieczenie żeby nie można było usunąć innych ważnych elementów przez przypadek

Przed usunięciem



I po usunięciu zamówienia nr z00001

oraz implementacja

```
function usunZamowienie(){
        var input=document.getElementById("szukaj_zamowienie");
        if (/^(z[0-9]{5,})$/.test(input.value)==true){
                var element=document.getElementById(input.value);
                element.parentNode.removeChild(element);
                element=document.getElementById(input.value+'p');
                element.parentNode.removeChild(element);
        }else{
                alert("Hola hola tylko zamówienia");
        }
}
```