Damian Nowak is a CEO at Virtkick. He's a Ruby coder, an Arch Linux hacker, and drinks good beer.

XML Schema December 2007

Jakiś czas temu w ramach zajęć z Metod Reprezentacji Informacji na studiach musiałem się zaznajomić z szeroko chwalonym przez wykładowców standardem XML. O ile samo tworzenie własnego pliku XML wielką filozofią nie jest, ponieważ to użytkownik tworzy sobie strukturę i znaczniki, o tyle XML Schema nie jest już dowolna. Stworzenie jej za pierwszym razem zajmuje trochę czasu, ponieważ trzeba zaznajomić się z wszystkimi dostępnymi elementami oraz wypróbać to w praktyce.

Czym w ogóle jest enigmatyczna XML Schema? XML Schema jest zbiorem zasad, jakimi należy się kierować przy wypełnianiu tworzonego przez nas pliku XML.

Przykład 1. Plik XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<pracownicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="pracownicy.xsd">
  <pracownik>
    <imie>Alojzy</imie>
    <nazwisko>Filipski</nazwisko>
    <rocznik>1967</rocznik>
    </pracownik>
    <pracownik>
    <imie>Dawid</imie>
    <nazwisko>Raskolnikow</nazwisko>
    <rocznik>1982</rocznik>
  </pracownik>
</pracownicy>
```

Atrybuty `xmlns:xsi` oraz `xsi:noNamespaceSchemaLocation` informują, że w pliku `pracownicy.xsd` znajduje się zbiór zasad dla tego pliku XML i jest on w formacie XML Schema.

Przykład 2. Plik XML Schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="pracownicy">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="pracownik" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="pracownik">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="imie" type="xs:string"/>
        <xs:element name="nazwisko" type="xs:string"/>
        <xs:element name="rocznik" type="xs:short"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Powyższy zapis należy interpretować w następujący sposób:

- Element “pracownicy” jest rodzicem dla sekwencji elementów “pracownik”,
- Wewnątrz elementu pracownik znajdują się elementy “imie”, “nazwisko”, “rocznik” oraz występują tylko jeden raz w ramach elementu “pracownik”,
- W elementach “imie” oraz “nazwisko” mogą znajdować się dowolne ciągi znaków,
- W elemencie “rocznik” może znajdować się krótka liczba całkowita.

Nazewnictwo

Wszystko rozjaśni zapis jeden przykład:

```
<element atrybut="wartość atrybutu">wartość elementu</element>
```

W HTML-u przyjęła się nazwa “znacznik”. “Znacznik” oznacza tak naprawdę “element”.

XML Schema - opis elementów

xs:simpleType

Określa, że wartość jest… tylko jedną wartością. Łatwiej powiedzieć więc, czym xs:simpleType nie jest - mianowicie w xs:simpleType nie może istnieć zagnieżdżanie kolejnych elementów np.:

```
<przyklad> <a>a</a> <b>b</b> </przyklad>
```

`xs:SimpleType` to jedynie coś takiego:

```
<przyklad>a b c</przyklad>
```

Na `xs:SimpleType` można nakładać ograniczenia poprzez `xs:restriction`.

xs:restriction

`<xs:restriction base="xs:typ"> ... </xs:restriction>` Określa ograniczenia dla elementów.

```
<xs:minLength value="8"/>
<xs:maxLength value="20"/>
```

Minimalne oraz maksymalne długości.

```
<xs:enumeration value="mężczyzna"/>
<xs:enumeration value="kobieta"/>
```

Wartość może przybrać tylko jedną z wartości.

```
<xs:minInclusive value="1600"/>
<xs:maxInclusive value="2010"/>
```

Wartość liczbowa musi być większa od 1600 i mniejsza od 2010.

```
<xs:pattern value="\d{2}-\d{2}-\d{5}-\d{1}"/>
```

Wartość może być tylko i wyłącznie postaci XX-XX-XXXXX-X, gdzie X to cyfry 0-9. Czytaj: wyrażenia regularne.

Przykład. `xs:simpleType`

```
<xs:element name="kod_pocztowy">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="6"/>
      <xs:maxLength value="6"/>
      <xs:pattern value="\d{2}-\d{3}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Wartości elementu “kod_pocztowy” muszą być typu string o długości równej 6. Wartość musi być postaci XX-XXX, gdzie X to cyfry.

xs:complexType

ComplexType to przeciwieństwo xs:SimpleType. Wewnątrz mogą się znajdować kolejne elementy.

xs:sequence

Określa jaka sekwencja elementów musi się znaleźć wewnątrz pewnego elementu. Określa również, jakie atrybuty posiada element.

```
<xs:sequence>
  <xs:element name="a" type="xs:string"/>
  <xs:element name="b" type="xs:string"/>
</xs:sequence>
```

>Przykład. xs:complexType

```
<xs:element name="zagadnienie">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="nazwa"/>
      <xs:element ref="bibliografia" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="typ" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="pojęcie"/>
          <xs:enumeration value="wzór chemiczny"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

Dla uproszczenia zapisu, zamiast opisywać każdy element w tym miejscu zastosowałem referencję - atrybut “ref”. Oznacza to, że znacznik zostanie scharakteryzowany w dalszej części pliku XML Schema, np. w taki sposób:

```
<xs:element name="nazwa" type="xs:string"/>
<xs:element name="bibliografia" ... >
```

`minOccurs`, `maxOccurs` - ilość wystąpień danego elementu. Liczby albo unbounded.

`xs:attribute` - określa jakie atrybuty może posiadać element. Use: required oznacza, że dany atrybut musi zawsze występować przy elemencie; use: optional, gdy może, ale nie musi.

Fragment. XML spełniający kryteria dla xs:complexType.

```
<zagadnienie typ="pojęcie">
  <nazwa>...</nazwa>
  <bibliografia>...</bibliografia>
</zagadnienie>
<zagadnienie typ="wzór chemiczny">
  <nazwa>...</nazwa>
  <bibliografia>...</bibliografia>
</zagadnienie>
```

XML Schema w praktyce

Z doświadczenia wiem, że najłatwiejszą formą nauki XML-podobnych języków (HTML, XML Schema, XSLT) jest analiza gotowego dokumentu XML. Zamieszczam więc cały plik XML wraz z XML Schemą, który stanowił zaliczenie projektu z Metod Reprezentacji Informacji.

- XML Instance
- XML Schema

Licencja użycia: użycie plików XML jest dozwolone wyłącznie w celach edukacyjnych. Oznacza to, że publikowanie plików XML bez mojej wiedzy i zgody jest zabronione.

Przydatne zasoby:

- Wikipedia: XML

- Wikipedia: XML Schema
- Walidator plików XML na podstawie dostarczonej XML Schema
- Altova XMLSpy - generowanie XML Schema na podstawie pliku XML*
- warto sobie wygenerować, jeśli mamy kłopot z prawidłowym napisaniem pliku XML Schema. Taka Schema nie jest jednak idealna, tzn. jest trochę chaotyczna i nagminnie używa enumeration. Niemniej jednak, plik XML jest poprawny według tak wygenerowanej XML Schema.

Damian Nowak
CEO & Ruby Developer