Exercises: XSD, XPath

Basi di dati 2

Disheng Qiu
disheng.qiu@gmail.com

Luca Rossi
luca.rossi.917@gmail.com

Hints:

- Use a validator
- XSD
  - **Eclipse** has a plugin for XML/XSD/DTD validation
  - **W3C Validator**: http://www.w3.org/2001/03/webdata/xsv
  - **Another Validator:** http://www.xmlforasp.net/schemavalidator.aspx
- XPath
  - Guess what? There's a plugin for **Eclipse**
  - Extensions for **Firefox** and **Chrome**

Exercises:

1x From XSD to instance

2x From instance to XSD

1x From specs to XSD

Exercise 1: *Carattere*

From XSD to instance

Ex.1 - From XSD to instance: *Carattere*

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd=”
        http://www.w3.org/2001/XMLSchema">
<xsd:simpleType name="cifra">
        <xsd:restriction base="xsd:string">
                <xsd:pattern value="[0-9]"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="carattere">
        <xsd:restriction base="xsd:string">
                <xsd:pattern value="[A-Z]"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:element name="car" type="carattere"/>
<xsd:element name="num" type="cifra"/>
<xsd:element name="esercizio" type="Tesercizio"/>
<xsd:complexType name="Tesercizio">
        <xsd:sequence>
                <xsd:element ref="car" minOccurs="6"
                                        maxOccurs="6"/>
                …
                <xsd:element ref="num" minOccurs="2"
                                        maxOccurs="2"/>
                <xsd:element ref="car"/>
                <xsd:element ref="num" minOccurs="2"
                                        maxOccurs="2"/>
                <xsd:element ref="car"/>
                <xsd:element ref="num" minOccurs="3"
                                        maxOccurs="3"/>
                <xsd:element ref="car"/>
        </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Ex.1 - From XSD to instance: *Carattere*

```
<xsd:simpleType name="cifra">
        <xsd:restriction base="xsd:string">
                <xsd:pattern value="[0-9]"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="carattere">
        <xsd:restriction base="xsd:string">
                <xsd:pattern value="[A-Z]"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:element name="car" type="carattere"/>
```

```
…
<xsd:element ref="num" minOccurs="2"
                       maxOccurs="2"/>
<xsd:element ref="car"/>
<xsd:element ref="num" minOccurs="3"
                       maxOccurs="3"/>
<xsd:element ref="car"/>
    </xsd:sequence>
</xsd:complexType>
```

Many “styles” of XSD:

- **Anonymous** vs **Nominal** types
- **Local** vs **Global** elements

Ex.1 - Solution

```
<esercizio>
    <car>X</car><car>X</car><car>X</car>
    <car>Y</car><car>Y</car><car>Y</car>
    <num>1</num><num>1</num>
    <car>Z</car>
    <num>2</num><num>2</num>
    <car>H</car>
    <num>3</num><num>3</num><num>3</num>
    <car>K</car>
</esercizio>
```

Exercise 2: *Binario*

From specs to XSD

Ex.2 - From specs to XSD: Binario

- Define a XSD that validates XML documents describing binary strings.

- The root element <binario> contains an arbitrary sequence of elements <uno> and <zero>, in any order.

Ex.2 - Solution

```
<xsd:element name="binario">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="zero" type="xsd:unsignedByte" fixed="0"/>
            <xsd:element name="uno" type="xsd:unsignedByte" fixed="1"/>
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
```

```
<binario>
    <uno>1</uno> <zero>0</zero> <zero>0</zero> <uno>1</uno> …
</binario>
```

Exercise 3: *Lettera*

From instance to XSD

Ex.3 - From instance to XSD: Lettera

```
<lettera>
    Gentile <cliente> tal dei tali </cliente>,
    la informiamo che i seguenti articoli da lei ordinati non sono più disponibili a magazzino:
    <ordine num="1234">
        <articolo>
            <codice>1</codice>
            <descr>articolo 1</descr>
        </articolo>
        <articolo>
            <codice>5</codice>
            <descr>articolo 5</descr>
        </articolo>
    </ordine>
    Cordiali saluti,
    <responsabile><tit>dr.</tit>Mario Rossi</responsabile>
</lettera>
```

Ex.3 – Solution (Global elements, nominal types)

```
<xsd:schema xmlns... >
<xsd:element name="lettera" type="Tlettera"/>
<xsd:element name="cliente" type="xsd:string"/>
<xsd:element name="ordine" type="Tordine"/>
<xsd:element name="articolo" type="Tarticolo" />
<xsd:element name="codice" type="xsd:string"/>
<xsd:element name="descr" type="xsd:string"/>
<xsd:element name="responsabile" type="Tresponsabile"/>
<xsd:element name="tit" type="xsd:string" />
<xsd:attribute name="num" type="xsd:integer"/>
<xsd:complexType name="Tlettera" mixed="true">
    <xsd:sequence>
        <xsd:element ref="cliente" /><xsd:element ref="ordine"/>
        <xsd:element ref="responsabile"/>
    </xsd:sequence>
</xsd:element>
```

```
<xsd:complexType name="Tordine" >
    <xsd:sequence>
        <xsd:element ref="articolo" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute ref="num"/>
</xsd:complexType>
<xsd:complexType name="Tarticolo">
    <xsd:sequence>
        <xsd:element ref="codice"/><xsd:element ref="descr"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Tresponsabile" mixed="true" >
    <xsd:choice minOccurs="0"><xsd:element ref="tit"/></xsd:choice>
</xsd:complexType>
```

Exercise 4: *Catena Montuosa*

From instance to XSD

Ex.4 - From instance to XSD: Catena Montuosa

```
<catenaMontuosa>
    <monte>
    <nome> Monte Bianco </nome>
        <regione> Valle d'Aosta </regione>
        <altezza unitaMisura="metri">4810</altezza>
    </monte>
    <monte>
        <nome>Gransasso</nome>
    </monte>
</catenaMontuosa>
```

Ex.4 – Solution 1 (Local elements, anonymous types)

```
<xsd:schema xmlns... >
<xsd:element name="catenaMontuosa" maxOccurs="unbounded">
 <xsd:complexType><xsd:sequence>
   <xsd:element name="monte" maxOccurs="unbounded"/>
    <xsd:complexType><xsd:sequence>
        <xsd:element name="nome" type="xsd:string"/>
        <xsd:element name="regione" type="xsd:string"/>
        <xsd:element name="altezza">

            <xsd:complexType>
             <xsd:simpleContent>
                 <xsd:extension base="xsd:integer">
                   <xsd:attribute name="unitaMisura" type="xsd:string"/>
                 </xsd:extension>
             </xsd:simpleContent>
            </xsd:complexType>

        </xsd:element>
    </xsd:sequence></xsd:complexType>
   </xsd:element>
 </xsd:sequence></xsd:complexType>
</xsd:element>
```

Ex.4 – Solution 2 (Global elements, anonymous types)

```
<xsd:element name="catenaMontuosa" maxOccurs="unbounded">
   <xsd:complexType><xsd:sequence>
        <xsd:element ref="monte" maxOccurs="unbounded"/>
   </xsd:sequence></xsd:complexType>
</xsd:element>
<xsd:element name="monte" >
   <xsd:complexType><xsd:sequence>
        <xsd:element ref="nome"/>
        <xsd:element ref="regione" minOccurs="0"/>
        <xsd:element ref="altezza" minOccurs="0"/>
   </xsd:sequence></xsd:complexType>
</xsd:element>

<xsd:element name="nome" type="xsd:string"/>
<xsd:element name="regione" type="xsd:string"/>
<xsd:element name="altezza" >
   <xsd:complexType>
     <xsd:simpleContent>
      <xsd:extension base="xsd:integer">
       <xsd:attribute name="unitaMisura" type="xsd:string"/>
      </xsd:extension>
     </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
```

Exercise 5: *XPath*

Ex.5 - DTD

```
<!ELEMENT recipes      (recipe*)>
<!ELEMENT recipe       (title, ingredient+, preparation, comment?, nutrition)>
<!ELEMENT title        (#PCDATA)>
<!ELEMENT ingredient   (ingredient*,preparation?)>
<!ATTLIST ingredient
    name               CDATA #REQUIRED
    amount             CDATA #IMPLIED
    unit               CDATA #IMPLIED>
<!ELEMENT preparation (step+)>
<!ELEMENT step         (#PCDATA)>
<!ELEMENT nutrition    EMPTY>
<!ELEMENT comment      (#PCDATA)>
<!ATTLIST nutrition
    calories           CDATA #REQUIRED
    fat                CDATA #REQUIRED
    carbohydrates      CDATA #REQUIRED
    protein            CDATA #REQUIRED
    alcohol            CDATA #IMPLIED>
```

Ex.5 - Instance

```
<recipes>
    <recipe>
      <title>Beef Parmesan</title>
      <ingredient name="beef" amount="1.5" unit="pound"/>
      <ingredient name="onion" amount="1"/>
      <ingredient name="green rings" amount="1"/>
      <preparation>
                <step>Boil pasta</step>
      </preparation>
      <comment>… </comment>
      <nutrition calories="1167" fat="23"carbohydrates="45" protein="32"/>
    </recipe>
 ….
</recipes>
```

Ex.5 – XPath Queries

1. "The titles of all recipes that use olive oil.”
2. "The titles of all recipes, returned as strings.”
3. "The titles of all recipes with less than 500 calories.”
4. "The amount of sugar needed for Zuppa Inglese.”
5. "The first two steps in preparing Zuppa Inglese.”
6. " The titles of all recipes that requires five steps.”
7. "The recipes that have an ingredient in common with Zuppa Inglese.“
8. "The titles of recipes have some compound ingredients."

Ex.5 - Solution

1. "The titles of all recipes that use olive oil.”

    - //recipe[ingredient/@name="olive oil"]/title

Ex.5 - Solution

1. "The titles of all recipes that use olive oil."

   - //recipe[ingredient/@name="olive oil"]/title NO
     (Attenzione agli ingredienti nidificati)

   - //recipe[.//ingredient/@name="olive oil"]/title
     (Vogliamo ottenetere titoli come ELEMENTI o come testo?) SI

   - //recipe[.//ingredient/@name="olive oil"]/title/text() SI

Ex.5 - Solution

1. "The titles of all recipes that use olive oil.”
2. "The titles of all recipes, returned as strings.”

    - //title/text()
      (è possibile specificare il documento tramite fn:doc("recipes.xml"))

Ex.5 - Solution

1. "The titles of all recipes that use olive oil.”
2. "The titles of all recipes, returned as strings.”
3. "The titles of all recipes with less than 500 calories.”

    - //recipe[//@calories < 500]/title

Ex.5 - Solution

1. "The titles of all recipes that use olive oil."
2. "The titles of all recipes, returned as strings."
3. "The titles of all recipes with less than 500 calories."

- //recipe[//@calories < 500]/title NO
  (Bisogna specificare l'attributo di quale elemento - // nelle parantesi torna da root)

- //recipe[nutrition/@calories < 500]/title SI

Ex.5 - Solution

1. "The titles of all recipes that use olive oil.”
2. "The titles of all recipes, returned as strings.”
3. "The titles of all recipes with less than 500 calories.”
4. "The amount of sugar needed for Zuppa Inglese.”

   - number(//recipe[title="Zuppa Inglese"]//ingredient[@name="sugar"]/@amount) (ritorna un numerico dell’ammontare dello zucchero – NON ESEGUE LA SOMMA)

Ex.5 - Solution

1. "The titles of all recipes that use olive oil.”
2. "The titles of all recipes, returned as strings.”
3. "The titles of all recipes with less than 500 calories.”
4. "The amount of sugar needed for Zuppa Inglese.”
5. "The first two steps in preparing Zuppa Inglese.”

- //recipe[title="Zuppa Inglese"]/preparation/step[position()=1 or position()=2]

Ex.5 - Solution

1. "The titles of all recipes that use olive oil.”
2. "The titles of all recipes, returned as strings.”
3. "The titles of all recipes with less than 500 calories.”
4. "The amount of sugar needed for Zuppa Inglese.”
5. "The first two steps in preparing Zuppa Inglese.”
6. " The titles of all recipes that requires five steps.”

Ex.5 - Solution

1. "The titles of all recipes that use olive oil.”
2. "The titles of all recipes, returned as strings.”
3. "The titles of all recipes with less than 500 calories.”
4. "The amount of sugar needed for Zuppa Inglese.”
5. "The first two steps in preparing Zuppa Inglese.”
6. " The titles of all recipes that requires five steps.”

- //recipe[.//step[5]]/title
- //recipe[preparation/step[5]]/title

Ex.5 - Solution

1. "The titles of all recipes that use olive oil.”
2. "The titles of all recipes, returned as strings.”
3. "The titles of all recipes with less than 500 calories.”
4. "The amount of sugar needed for Zuppa Inglese.”
5. "The first two steps in preparing Zuppa Inglese.”
6. " The titles of all recipes that requires five steps.”
7. "The recipes that have an ingredient in common with Zuppa Inglese.“

- //recipe[.//ingredient/@name=//recipe[title="Zuppa Inglese"]//ingredient/@name]

Ex.5 - Solution

1. "The titles of all recipes that use olive oil."
2. "The titles of all recipes, returned as strings."
3. "The titles of all recipes with less than 500 calories."
4. "The amount of sugar needed for Zuppa Inglese."
5. "The first two steps in preparing Zuppa Inglese."
6. " The titles of all recipes that requires five steps."
7. "The recipes that have an ingredient in common with Zuppa Inglese."
8. "The titles of recipes have some compound ingredients."

- //recipe[ingredient/ingredient]/title