Theme 3: XML Schema

Introduction & Simple Types

XML Schema Basics

- XML Schema is a **schema language**, like DTD.

- The W3C developed it to address DTD's shortcomings:
  - It's a **custom XML language**;
  - You can define **data types** for element/attribute values,
  - It supports **namespaces**, and
  - You can define **local** and **global** elements/attributes.

Working with XML Schema

- In XML Schema, you must define an element/attribute...
- ...in terms of a specific **data type**.

- **The two major kinds of data types in XML Schema:**
  - Simple types, and
  - Complex types.

Working with XML Schema

- A **simple type** element can contain only text. no attributes, no children, just text
- A **complex type** element can contain one of the following:
  - **Child elements** and/or **attributes**;
  - **Mixed data** and/or **attributes**;
  - **Text** and **attributes**, or
  - **Nothing** (empty) and/or **attributes**.

Beginning an XML Schema

- An XML Schema doc is a text file with extension, **.xsd**.
- **XSD** stands for **X**ML **S**chema **D**efinition.

- The outer structure of an XSD doc must look like this:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <!-- The schema rules -->
</xs:schema>
```

Linking the Schema to the XML

- To validate an XML doc using an XSD doc...
- ...you must add a **link** in the XML doc.

- Add **two attributes** to the XML's root to create the link:

```
<theRoot
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="filename.xsd">
```

Linking the Schema to the XML

- Once correctly linked, you can use **EditiX**...
- ...to validate the XML against an XSD doc.

- Ensure that all files are in the correct folders...
- ...open the XML doc in **EditiX**, then press **Ctrl+K**...
- ...to check if the XML is well-formed and valid.

Annotating Schemas

- You can use standard XML comments in an XSD doc.

- Alternatively, you can use **annotations**. sometimes data actually used and not just ignored like comments

- The processor ignores standard XML comments...
- ...but it can parse annotations.

- Use annotations if you need their info during processing.

Annotating Schemas

- **To annotate XML Schemas:**

```
<xs:annotation>
        <xs:documentation>
                <!-- Some notes -->
        </xs:documentation>
</xs:annotation>
```

- You can add this anywhere inside the schema’s root.

Simple Types

Defining Simple Types

- **Reminder:** A simple type element can contain only text.
- The specific kind of text depends on the simple type.
- There are built-in types for the most common kinds, e.g.:
  - Strings, Booleans, URLs, dates, times, numbers, etc.
- You can limit the built-in types further using **facets**.

Defining Simple Types

- A **facet** is a restriction, e.g.:
  - The element may contain only strings, **but**...
  - ...the string length must be at most 10 characters.

- When you apply **facets** to an existing simple type...
- ...you are creating a **custom simple type**.

Simple Type Elements

- **To define a simple type element in the XSD doc:**

```
<xs:element name="el_name" type="xs:string"/>
```

- Replace **el_name** with the name of the element.

- XML Schema-specific elements must have the prefix, **xs:**.

- Referencing a built-in type must also be preceded by **xs:**.

Simple Type Elements

- Instead of **xs:string**, you can also use:
  - **xs:decimal**/**xs:integer**/**xs:float** for numbers;
  - **xs:boolean** for “true” or “false”, “1” or “0”;
  - **xs:date** for dates;
  - **xs:time** for times;
  - **xs:anyURI** for references to files or web locations;
  - Or any of the other built-in simple types.

- Find the full list of built-in simple types here:

  www.w3.org/TR/xmlschema-2/#built-in-datatypes

Simple Type Elements – Date and Time

- **The available built-in date and time simple types are:**
  - **xs:date** for dates in **YYYY-MM-DD** format.
  - **xs:time** for times in **hh:mm:ss** format.
  - **xs:dateTime** for a combo: **YYYY-MM-DDThh:mm:ss**.
    - The date and time are separated with a **T** (for **T**ime).

Simple Type Elements – Period

- **xs:duration** for an amount of time.
  - Duration is formatted like this: **PnYnMnDTnHnMnS**.
  - **P** (Period) is required.
  - **T** and its following characters are only needed if you have time units.
  - **Y** is for years, **M** for months, **D** for days, **H** for hours...
  - ...**M** for minutes, and **S** for seconds.
  - Each **n** is the number of each unit.
  - E.g. 3 months, 4 days, 6 hours, and 17 minutes = **P3M4DT6H17M**
  - Add a minus (**-**) in front if it is a **past** duration (instead of future).
  - E.g. 90 days ago = **-P90D**.

Simple Type Elements – Date

- **xs:gYear** for a year value in **YYYY** format (**g** = Gregorian).
- **xs:gYearMonth** for a year and a month: **YYYY-MM**.
- **xs:gMonth** for a month: **--MM** (note: two dashes).
- **xs:gMonthDay** for a month and day: **--MM-DD**.
- **xs:gDay** for a day: **---DD** (note: three dashes).

Simple Type Elements – Number Types

- Some built-in simple types for numbers:
  - **xs:decimal** for numbers that may have a decimal point.
  - **xs:integer** for whole numbers.
  - **xs:positiveInteger** for positive whole numbers. Similarly:
    - **xs:negativeInteger**,
    - **xs:nonPositiveInteger**, and
    - **xs:nonNegativeInteger**.

  - **xs:float** for floating point numbers.

Predefining Element Content

- You can set a **fixed** or **default** value for an element.

only default get added to element if it's left out, if there is a fixed it will not be added

- **To set a fixed value:**

```
<xs:element name="el_name" type="xs:string" fixed="value"/>
```

- **Fixed** means that if **el_name** appears in the XML...
- ...it **must** have the fixed value.

Predefining Element Content

- If **el_name** appears in the XML and it is empty...
- ...the processor gives it the fixed value.

- If **el_name** is omitted from the XML...
- ...the processor **will not** add it automatically.

Predefining Element Content

- **To set a default value:**

```
<xs:element name="el2_name" type="xs:string" default="value"/>
```

- If **el2_name** appears in the XML, but it is empty...
- ...the processor gives it the default value.

Predefining Element Content

- If **el2_name** is omitted from the XML...
- ...the processor adds it with the default value.

- **el2_name** may appear in the XML with a different value.

- You cannot use **fixed** and **default** at the same time!

Custom Types

Custom Simple Types

- Create custom types by restricting an existing type:

```
<xs:element name="el_name">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <!-- One or more facets -->
            </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="el_name" type="xs:string">
```

Deriving Custom Simple Types

- A custom type created within an element definition...
- ...is called an **anonymous custom type.** cannot reuse

differene: reusablility

- Alternatively, you can create a **named custom type.**
  - It is created separately from the element definition...
  - ...so that you can reuse it for many elements.

Deriving Custom Simple Types

- An example using a **named custom type**:

instread of defining element, you defien simpleType.... can reference later in doc

```
<xs:simpleType name="type_name">
        <xs:restriction base="xs:string">
            <!-- One or more facets -->
        </xs:restriction>
</xs:simpleType>

<xs:element name="el_name" type="type_name"/>
```

Deriving Custom Simple Types

- If your type is **named**, you can give it any **type_name**.
- When referring to a custom type, don't add **xs:**.
- You can use any built-in type in the place of **xs:string**.
- You can even use an existing custom type as foundation...
- ...for a new custom type with even more restrictions.

Facets

- To define a custom simple type, you must...
- ...restrict an existing simple type using **facets**. google: API on what facets you can use

- Facets are XML Schema tags inside **xs:restriction**.
  - E.g. **xs:maxInclusive**
  - **xs:enumeration**, etc.

Facets

- **E.g. if the value must be an integer <= 6856:**

```
<xs:element name="total_bases">
        <xs:simpleType>
            <xs:restriction base="xs:integer">
                <xs:maxInclusive value="6856"/>
            </xs:restriction>
        </xs:simpleType>
</xs:element>
```

Facets

- **E.g. if the value must be between two boundaries:**

```
<xs:element name="game_day">
        <xs:simpleType>
            <xs:restriction base="xs:date">
                <xs:minExclusive value="1854-04-13"/>
                <xs:maxExclusive value="1976-10-03"/>
            </xs:restriction>
        </xs:simpleType>
</xs:element>
```

Facets

- **E.g. if the value must be a choice between values:**

```
<xs:element name="wonder_name">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="Colossus of Rhodes"/>
            <xs:enumeration value="Pyramid of Giza"/>
            <xs:enumeration value="Statue of Zeus"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Facets

- **E.g. if the value must be a string of 5 characters:**

```
<xs:element name="wonder_code">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:length value="5"/>
            </xs:restriction>
        </xs:simpleType>
</xs:element>
```

Facets

- **E.g. a string with at most 256 characters:**

```
<xs:element name="brief_description">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:maxLength value="256"/>
            </xs:restriction>
        </xs:simpleType>
</xs:element>
```

Facets

- **E.g. if it must have 6 digits, 4 of them after the decimal point (e.g. 25.2324):**

```
<xs:element name="atomic_weight">
        <xs:simpleType>
            <xs:restriction base="xs:decimal">
                <xs:totalDigits value="6"/>
                <xs:fractionDigits value="4"/>
            </xs:restriction>
        </xs:simpleType>
</xs:element>
```

Facets

- You can use the **xs:pattern** facet...
- ...to make a value match a **regular expression**.

- We won't cover regular expressions in IMY 210...
- ...but you can read more about them in the textbook.

- See also: http://www.w3.org/TR/xmlschema-0/#SimpleTypeFacets

Deriving a List Type

- You can derive a **list type** from an existing simple type.

- A list type element can contain a list of values...
- ...with each value separated by a space.

- Create an **anonymous** list type for using it once...
- ...or a **named** list type if you want to reuse it.

Deriving a List Type

- **E.g. a named list type derived from** xs:date**:**

```
<xs:simpleType name="date_list">
          <xs:list itemType="xs:date"/>
    </xs:simpleType>
...
<xs:element name="eclipses" type="date_list"/>
```

Then, in the XML:

```
<eclipses>2010-12-21 2011-06-15 2011-12-10</eclipses>
```

Deriving a Union Type

- You can derive a **union type** from...
- ...a collection of existing simple types.

- A union type element must have a **single value**...
- ...whose data type must be one of the types in the union.

e.g. A value can either be a **date** OR a **date and time** value

Deriving a Union Type

- **E.g. an element rule with an anonymous union type:**

```
<xs:element name="when">
        <xs:simpleType>
            <xs:union memberTypes="xs:date xs:dateTime"/>
        </xs:simpleType>
</xs:element>
```

- The **when** element must have a **single value** that can either be of type **date** or of type **dateTime**.

Theme 3: XML Schema

- TO BE CONTINUED…