Essential skills

Tykushin Anatoly, a.tykushin@innopolis.ru, MSIT-SNE

September 2016

Contents

1 Assignment 1.1 **1**
1.1 Installation of wiki engine 1
1.1.1 Choosing wiki engine 1
1.1.2 Installation process 1
1.2 Organizing easy navigation 2

2 Assignment 1.2 **3**
2.1 Creating entities 3
2.2 XML file with data 3
2.3 DTD file 4
2.4 XSD file 4
2.5 Validating files 5

3 Assignment 1.3 **6**
3.1 XSLT stylesheet that transforms XML to XHTML 6
3.2 Writing CSS style 7
3.3 Checking validation of created files 8

1 Assignment 1.1

1.1 Installation of wiki engine

1.1.1 Choosing wiki engine

Dokuwiki was chosen as a wiki engine. Here is a couple of reasons:

- simply to use, high versatile
- open source
- doesn't require any database
- clean, readable syntax
- easy maintenance, backup and integration.

1.1.2 Installation process

Order of installation process has been presented in the list:

1. Update & Upgrade OS
2. Install Apache2 and php-lib (php7.0 and libapache2-mod-php). A couple of problems happened, needed to install php-xml library manually because the lack of ***utf8-(en/de)code()*** function.

3. Dowloading and decompressing Dokuwiki archive to the **/var/www/dokuwiki** directory
4. Configuting apache2 files:

   sites-enabled/000-*.conf to change root directory of apache

   apache2.conf port=80 $\rightarrow$ 11180, **ALLOWING OVERRIDE** option set
5. Finishing installation at hostname:11180/install.php, after that deleting install.php script

1.2 Organizing easy navigation

Using power of simplicity of Dokuwiki it was really easy to make a pretty startpage. You can go there using this link: dokuwiki-start-page

2 Assignment 1.2

2.1 Creating entities

In the diagram below you can see 4 entities, their fields, and dependencies between them:

```
                                                     .___________.
    .___________.                                    |   skin    |
    | zootopia  |        .___________.               |-----------|
    |---------- |        |   animal  |       +------ | colortype |
    |  animal   |<-------|-----------|       +       | palette   |
    |___________|        |     id    |       +       |___________|        .___________.
                         |    skin   |<------+                            |   noise   |
                         |   noise   |<-----------------------------------|-----------|
                         |    eyes   |<-------+                           |   voice   |
                         |___________|        +                           |  loudness |
                                              +                           |   range   |
                                              +                           |___________|
                                              +
                                              +
                                              +------.___________.
                                                     |   eyes    |
                                                     |-----------|
                                                     |   size    |
                                                     |   form    |
                                                     |___________|
```

- animal
- skin
- noise
- eyes

Entity zootopia is an aggregation type, which consists of many

2.2 XML file with data

Using scheme in the subsection 2.1 we can create sample file with data about animals. In order to keep report clean enough here you can see sample info about mockingbird. You can access the whole file here:

- XML file in dokuwiki
- XML file in GitHub

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="zootopia.xsl"?>
<zootopia>
    <animal id="4">
              <type>mockingbird</type>
             <skin>
                 <colortype>mixed</colortype>
                 <palette>bright</palette>
```

```
            </skin>
            <noise>
                <voice>chirp</voice>
                <loudness>mid</loudness>
                <range>soprano</range>
            </noise>
            <eyes>
                <size>tiny</size>
                <form>circular</form>
            </eyes>
    </animal>
</zootopia>
```

2.3 DTD file

DTD file created according to the chosen in subsection 2.1 schema.

```
<!DOCTYPE zootopia [
<!ELEMENT zootopia (animal+)>
<!ELEMENT animal (type,skin,noise,eyes)>
<!ATTLIST animal id CDATA #REQUIRED>
<!ELEMENT type (#PCDATA)>
<!ELEMENT skin (colortype,palette)>
<!ELEMENT noise (voice,loudness,range)>
<!ELEMENT eyes (size,form)>
<!ELEMENT colortype (#PCDATA)>
<!ELEMENT palette (#PCDATA)>
<!ELEMENT voice (#PCDATA)>
<!ELEMENT loudness (#PCDATA)>
<!ELEMENT range (#PCDATA)>
<!ELEMENT size (#PCDATA)>
<!ELEMENT form (#PCDATA)>
]>
```

2.4 XSD file

XSD file created according to the chosen in subsection 2.1 schema too. There are a couple of ways to organize XSD. The chosen design method is based on defining all elements and attributes first, and then referring to them using the ref attribute. This file is also compressed, full one can be accessed here:

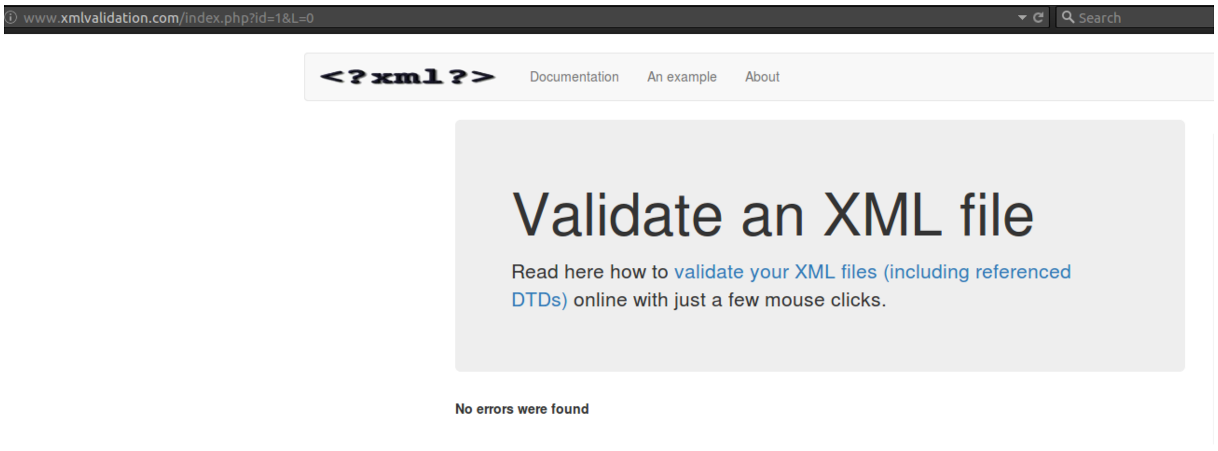
- XSD file in dokuwiki
- XSD file in GitHub

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

      <!-- Definition of attributes -->
    <xs:attribute name="id" type="xs:integer"/>

    <!-- Definition of simple elements -->
    <xs:element name="type" type="xs:string"/>
    <!-- Definition of other elements -->

    <!-- Definition of complex elements 1st level-->
    <xs:element name="skin">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="colortype"/>
          <xs:element ref="palette"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
   <!-- Definition of other elements -->

       <!-- Definition of complex elements 2nd level-->
    <xs:element name="animal">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="type"/>
          <xs:element ref="skin"/>
          <xs:element ref="noise"/>
          <xs:element ref="eyes"/>
        </xs:sequence>
        <xs:attribute ref="id" use="required"/>
      </xs:complexType>
    </xs:element>

    <!-- Definition of complex elements 3rd level-->
    <xs:element name="zootopia">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="animal" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>

</xs:schema>
```

2.5 Validating files

Validation check of dtd and xml is shown on the figure 1.

Validation check of xsd and xml is shown on the figure 2.

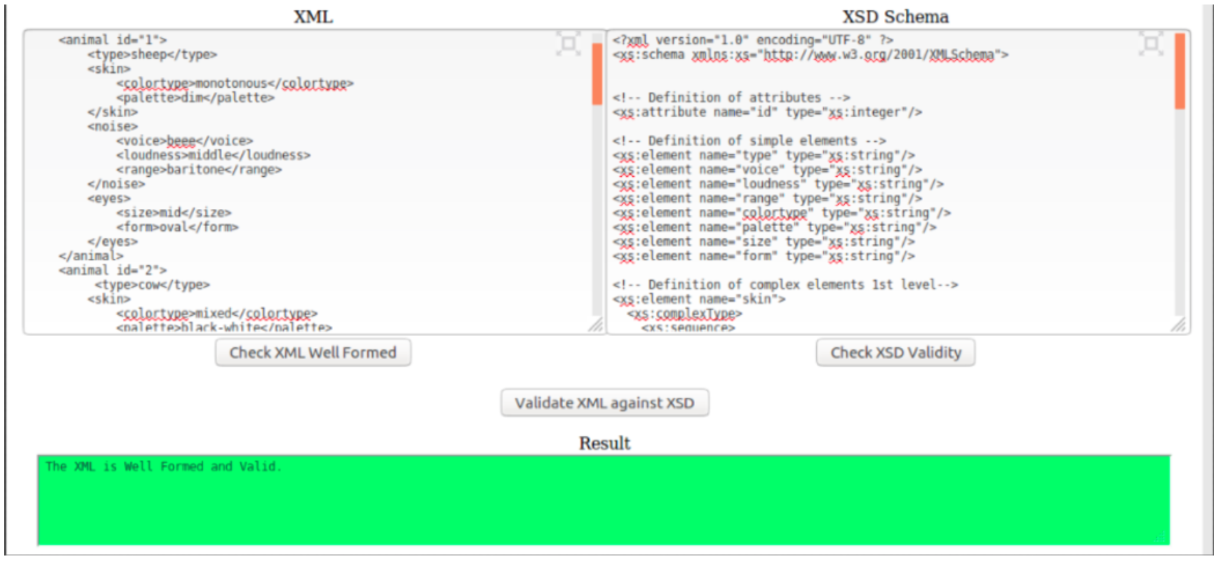


Validate an XML file

Read here how to validate your XML files (including referenced DTDs) online with just a few mouse clicks.

No errors were found

Figure 1: Checking validation



```
    <animal id="1">
        <type>sheep</type>
        <skin>
            <colortype>monotonous</colortype>
            <palette>dim</palette>
        </skin>
        <noise>
            <voice>beee</voice>
            <loudness>middle</loudness>
            <range>baritone</range>
        </noise>
        <eyes>
            <size>mid</size>
            <form>oval</form>
        </eyes>
    </animal>
    <animal id="2">
         <type>cow</type>
        <skin>
            <colortype>mixed</colortype>
            <palette>black-white</palette>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">


<!-- Definition of attributes -->
<xs:attribute name="id" type="xs:integer"/>

<!-- Definition of simple elements -->
<xs:element name="type" type="xs:string"/>
<xs:element name="voice" type="xs:string"/>
<xs:element name="loudness" type="xs:string"/>
<xs:element name="range" type="xs:string"/>
<xs:element name="colortype" type="xs:string"/>
<xs:element name="palette" type="xs:string"/>
<xs:element name="size" type="xs:string"/>
<xs:element name="form" type="xs:string"/>

<!-- Definition of complex elements 1st level-->
<xs:element name="skin">
  <xs:complexType>
    <xs:sequence>
```

```
The XML is Well Formed and Valid.
```

Figure 2: Checking validation

3 Assignment 1.3

3.1 XSLT stylesheet that transforms XML to XHTML

Contents of XSLT file is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
    <head>
      <link rel="stylesheet" href="zootopia.css" />
    </head>
  <body>
  <h2>My zootopia</h2>
  <table border="1">
```

```
    <tr>
      <th>ID</th>
      <th>Type</th>
      <th>Skin</th>
      <th>Noise</th>
      <th>Eyes</th>
    </tr>
    <xsl:for-each select="zootopia/animal">
    <tr>
      <td class='id-theme'><xsl:value-of select="@id"/></td>
      <td class='name-theme'><xsl:value-of select="type"/></td>
      <td class='skin-theme'><xsl:value-of select="skin/colortype"/>,
       <xsl:value-of select="skin/palette"/></td>
      <td class='noise-theme'>"<xsl:value-of select="noise/voice"/>",
       <xsl:value-of select="noise/loudness"/>,
       <xsl:value-of select="noise/range"/></td>
      <td class='eye-theme'><xsl:value-of select="eyes/size"/>,
       <xsl:value-of select="eyes/form"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

3.2 Writing CSS style

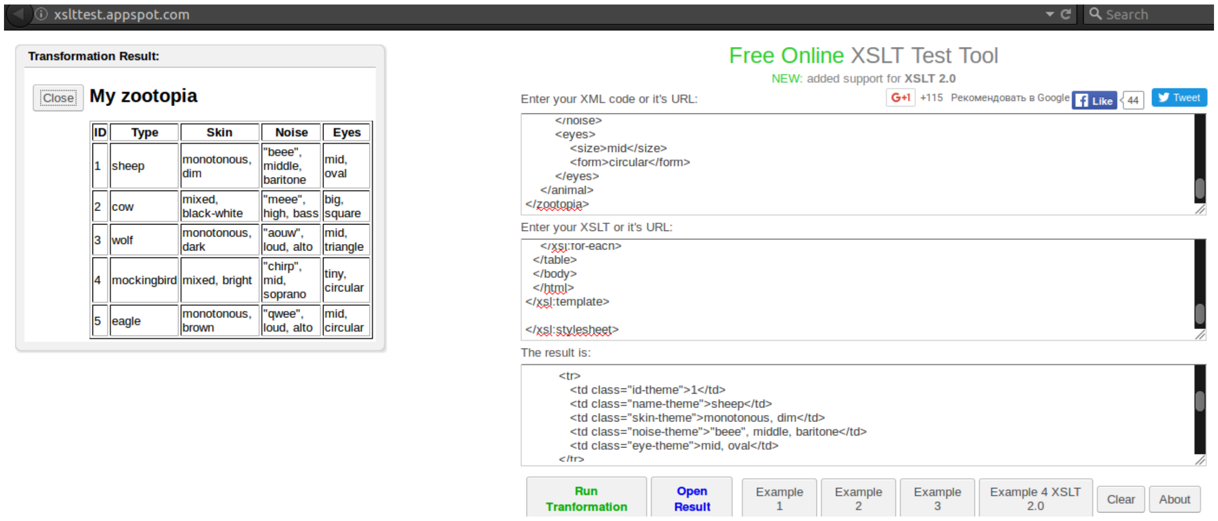
Contents of CSS file is shown below.

```
td {
    background: #99ccff;
}
th {
    background: #336699;
}
.id-theme {
    color: #0f0;
}
.name-theme:hover {
    color: #00f;
}
.voice-theme {
    color: #00f;
}
.voice-theme:hover {
    color: #0f0;
}
.skin-theme {
    color: #f00;
}
.skin:hover {
    color: #f0f;
}
```

```
.eye-theme {
    color: #f00;
}
.eye-theme:hover {
    color: #f0f;
}
```

3.3 Checking validation of created files

To make sure that everything works we just have to run zootopia.xml file. On the figure 3 is shown result of online validation check.



Transformation Result:

My zootopia

ID	Type	Skin	Noise	Eyes
1	sheep	monotonous, dim	"beee", middle, baritone	mid, oval
2	cow	mixed, black-white	"meee", high, bass	big, square
3	wolf	monotonous, dark	"aouw", loud, alto	mid, triangle
4	mockingbird	mixed, bright	"chirp", mid, soprano	tiny, circular
5	eagle	monotonous, brown	"qwee", loud, alto	mid, circular

Enter your XML code or it's URL:

```
        </noise>
        <eyes>
            <size>mid</size>
            <form>circular</form>
        </eyes>
    </animal>
</zootopia>
```

Enter your XSLT or it's URL:

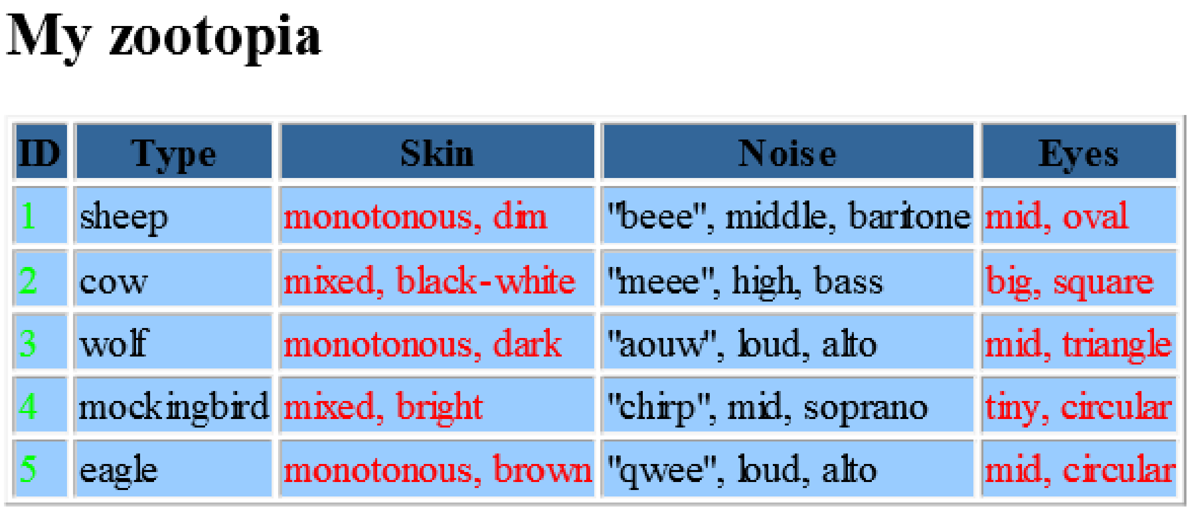
```
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

The result is:

```
         <tr>
            <td class="id-theme">1</td>
            <td class="name-theme">sheep</td>
            <td class="skin-theme">monotonous, dim</td>
            <td class="noise-theme">"beee", middle, baritone</td>
            <td class="eye-theme">mid, oval</td>
         </tr>
```

Figure 3: How XSLT works

On the figure 4 is shown how code was proceeded by the browser.



My zootopia

ID	Type	Skin	Noise	Eyes
1	sheep	monotonous, dim	"beee", middle, baritone	mid, oval
2	cow	mixed, black-white	"meee", high, bass	big, square
3	wolf	monotonous, dark	"aouw", loud, alto	mid, triangle
4	mockingbird	mixed, bright	"chirp", mid, soprano	tiny, circular
5	eagle	monotonous, brown	"qwee", loud, alto	mid, circular

Figure 4: How XSLT+CSS works