# Advanced Studies in IT CT433

## Exam Q&A

www.deri.ie

science foundation ireland
fondúireacht eolaíochta éireann

National University of Ireland, Galway
Ollscoil na hÉireann, Gaillimh

# XML

- Know what is well-formed XML, valid XML
- Well-formed:
  - Close opened tags and Pis
    - E.g. <br> is allowed in HTML, but not well-formed XML.

- Validity is always with respect to a grammar definition, given in
  - DTD, or
  - XML Schema

# DTD

- E.g.
  - Given a DTD and some XML fragments
    - Identify valid and invalid fragments

```
<!ELEMENT employees (marketing)>
<!ELEMENT marketing (employee+)>
<!ELEMENT employee (name,email+,tel*,fax?)>
<!ATTLIST employee id CDATA #IMPLIED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ELEMENT fax (#PCDATA)>
```

```
<employees>
  <marketing>
    <employee id="1834">
      <name>Gustav Sielmann</name>
      <email>gsielmann@Dot.com</email>
      <tel>+43/0662/723942-124</tel>
      <fax>+43/0662/723942-800</fax>
    </employee>
  </marketing>
</employees>
```

**ok**

```
   <employee id=”2222">
      <name>John</name>
      <name>Johnny</name>
      <email>gsielmann@Dot.com</email>
      <tel>+43/0662/723942-124</tel>
      <fax>+43/0662/723942-800</fax>
    </employee>
```

**Not ok**

```
   <employee id=”2222">
      <name>John</name>
      <email>gsielmann@Dot.com</email>
      <fax>+43/0662/723942-800</fax>
    </employee>
```

**ok**

# XSD

- See slides from assignments!

# Xpath, XSLT:

- Given some XML document and some simple queries in natural language, such as:

    - all courses of type "lecture" which took place in the year "2004"
    - the third course which has less than 2 exams
    - all titles of courses where the attribute sine-tempore is ”no"

- Be able to write down the respective XPath expressions, or identify the answers to such Xpath expressions.

# XSLT

- E.g., given the folllowing XML and XSLT, what is the resulting XML?

```
<?xml version="1.0"?>
      < xsl:stylesheet
xmlns:xsl=http://www.w3.org/1999/XSL/Transform
                       version="1.0"
xmlns="http://www.w3.org/1999/xhtml">

  <xsl:template match="/">
       <html xmlns="http://www.w3.org/1999/xhtml">
       <xsl:apply-templates/>
       </html>
  </xsl:template>

  <xsl:template match="card[@type='simple']">
       <xsl:apply-templates select="title"/>
       <body>
       <xsl:apply-templates select="name"/>
       <xsl:apply-templates select="email"/>
       </body>
   </xsl:template>

   <xsl:template match="card[not(@type='simple')]"/>

   <xsl:template match="name">
       <xsl:element name="h2">
       <xsl:attribute name="id">123</xsl:attribute>
       <xsl:value-of select="text()"/>
       </xsl:element>
   </xsl:template>

   <xsl:template match="email">
       <p>email: <a href="mailto:{text()}"><tt>
      < xsl:value-of select="text()"/>
       </tt></a></p>
   </xsl:template>

   <xsl:template match="title">
       <xsl:element name="title">
       <xsl:value-of select="text()"/>
      < /xsl:element>
   </xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0"?>
<businessCards>
   <card>
        <name>John Doe</name>
        <title>CEO, Widget Inc.</title>
        <email>john.doe@widget.com</email>
   </card>

   <card type="simple">
        <name>Max Muster</name>
        <title>Management Director, Widget Inc.</title>
        <email>max.muster@widget.com</email>
   </card>
</businessCards>
```

**Attention! The template matching non-simple cards, just ignores them. In case you made notes in the lecture today on what we had on the whiteboard, that might have been misleading!**

**?**

**Tip:**
**Try out this and the XSLT examples in the last few slides with Oxygen or XMLSpy and check the output to get a feeling!**

# RDF + SPARQL

- See last assignment sheet:
  - Given some natural language statements

    “alice knows john”, “john’s name is ‘John Doe’”, “alice’s homepage is http://www.alice.example.org/”, john’s email is doe.john@example.org
  - Be able to write them down as Turtle triples

- Write down some SPARQL queries then, e.g.
  - Persons who don’t have a homepage
  - Persons who know someone whose name is “John Doe”
  - Persons who know someone who knows alice
  - …

## More questions?

- Time allowed… some more XSLT examples…

# XSLT example 1:

A stylesheet for transforming documents that conform to a simple DTD into XHTML [XHTML]. The DTD is:

```
<!ELEMENT doc (title, chapter*)>
<!ELEMENT chapter (title, (para|note)*, section*)>
<!ELEMENT section (title, (para|note)*)>
<!ELEMENT title (#PCDATA|emph)*>
<!ELEMENT para (#PCDATA|emph)*>
<!ELEMENT note (#PCDATA|emph)*>
<!ELEMENT emph (#PCDATA|emph)*>
```

# XSLT Part 1

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns="http://www.w3.org/TR/xhtml1/strict">
<xsl:strip-space elements="doc chapter section"/>
<xsl:output  method="xml”  indent="yes"  encoding="iso-8859-1” />

<xsl:template match="doc">
 <html>
   <head>
     <title>
       <xsl:value-of select="title"/>
     </title>
   </head>
   <body>
     <xsl:apply-templates/>
   </body>
 </html>
</xsl:template>

<xsl:template match="doc/title">
  <h1>
    <xsl:apply-templates/>
  </h1>
</xsl:template>

<xsl:template match="chapter/title">
  <h2>
    <xsl:apply-templates/>
  </h2>
</xsl:template>

<xsl:template match="section/title">
  <h3>
    <xsl:apply-templates/>
  </h3>
</xsl:template>

</xsl:stylesheet>
```

.org

# XSLT Part 2

```
<xsl:template match="para">
  <p>
    <xsl:apply-templates/>
  </p>
</xsl:template>

<xsl:template match="note">
  <p class="note">
    <b>NOTE: </b>
    <xsl:apply-templates/>
  </p>
</xsl:template>

<xsl:template match="emph">
  <em>
    <xsl:apply-templates/>
  </em>
</xsl:template>
```

- Makes use of “standard template” which copy-outputs all text! (recall XSLT lecture).

# XSLT applied

```
<!DOCTYPE doc SYSTEM "doc.dtd">
<doc>
<title>Document Title</title>
<chapter>
<title>Chapter Title</title>
<section>
<title>Section Title</title>
<para>This is a test.</para>
<note>This is a note.</note>
</section>
<section>
<title>Another Section Title</title>
<para>This is <emph>another</emph> test.</para>
<note>This is another note.</note>
</section>
</chapter>
</doc>
```

```
<?xml version="1.0" encoding="iso-8859-1"?>
<html xmlns="http://www.w3.org/TR/xhtml1/strict">
<head>
<title>Document Title</title>
</head>
<body>
<h1>Document Title</h1>
<h2>Chapter Title</h2>
<h3>Section Title</h3>
<p>This is a test.</p>
<p class="note">
<b>NOTE: </b>This is a note.</p>
<h3>Another Section Title</h3>
<p>This is <em>another</em> test.</p>
<p class="note">
<b>NOTE: </b>This is another note.</p>
</body>
</html>
```

# XSLT another example

- three different XSLT stylesheets to produce three different representations of the data, HTML, SVG and VRML
- input data is:

```
<sales>

        <division id="North">
                <revenue>10</revenue>
                <growth>9</growth>
                <bonus>7</bonus>
        </division>

        <division id="South">
                <revenue>4</revenue>
                <growth>3</growth>
                <bonus>4</bonus>
        </division>

        <division id="West">
                <revenue>6</revenue>
                <growth>-1.5</growth>
                <bonus>2</bonus>
        </division>

</sales>
```

# The XSLT for SVG:

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns="http://www.w3.org/Graphics/SVG/SVG-19990812.dtd">

<xsl:output method="xml" indent="yes" media-type="image/svg"/>

<xsl:template match="/">

<svg width = "3in" height="3in">
    <g style = "stroke: #000000">
        <!-- draw the axes -->
        <line x1="0" x2="150" y1="150" y2="150"/>
        <line x1="0" x2="0" y1="0" y2="150"/>
        <text x="0" y="10">Revenue</text>
        <text x="150" y="165">Division</text>
        <xsl:for-each select="sales/division">
            <!-- define some useful variables -->
            <!-- the bar's x position -->
            <xsl:variable name="pos"
                          select="(position()*40)-30"/>
            <!-- the bar's height -->
            <xsl:variable name="height"
                          select="revenue*10"/>
            <!-- the rectangle -->
            <rect x="{$pos}" y="{150-$height}"
                  width="20" height="{$height}"/>
            <!-- the text label -->
            <text x="{$pos}" y="165">
                <xsl:value-of select="@id"/>
            </text>
            <!-- the bar value -->
            <text x="{$pos}" y="{145-$height}">
                <xsl:value-of select="revenue"/>
            </text>
        </xsl:for-each>
    </g>
</svg>

</xsl:template>
</xsl:stylesheet>
```

```
<svg width="3in" height="3in"
  xmlns="http://www.w3.org/Graphics/SVG/svg-19990412.dtd">
   <g style="stroke: #000000">
      <line x1="0" x2="150" y1="150" y2="150"/>
      <line x1="0" x2="0" y1="0" y2="150"/>
      <text x="0" y="10">Revenue</text>
      <text x="150" y="165">Division</text>
      <rect x="10" y="50" width="20" height="100"/>
      <text x="10" y="165">North</text>
      <text x="10" y="45">10</text>
      <rect x="50" y="110" width="20" height="40"/>
      <text x="50" y="165">South</text>
      <text x="50" y="105">4</text>
      <rect x="90" y="90" width="20" height="60"/>
      <text x="90" y="165">West</text>
      <text x="90" y="85">6</text>
   </g>
</svg>
```