

Semi-structured Data

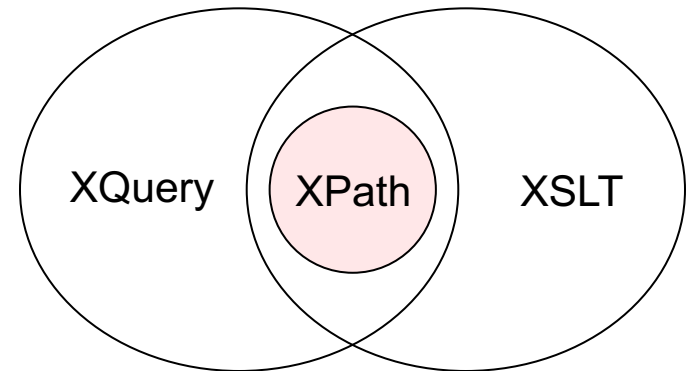
7 - XSLT

Outline

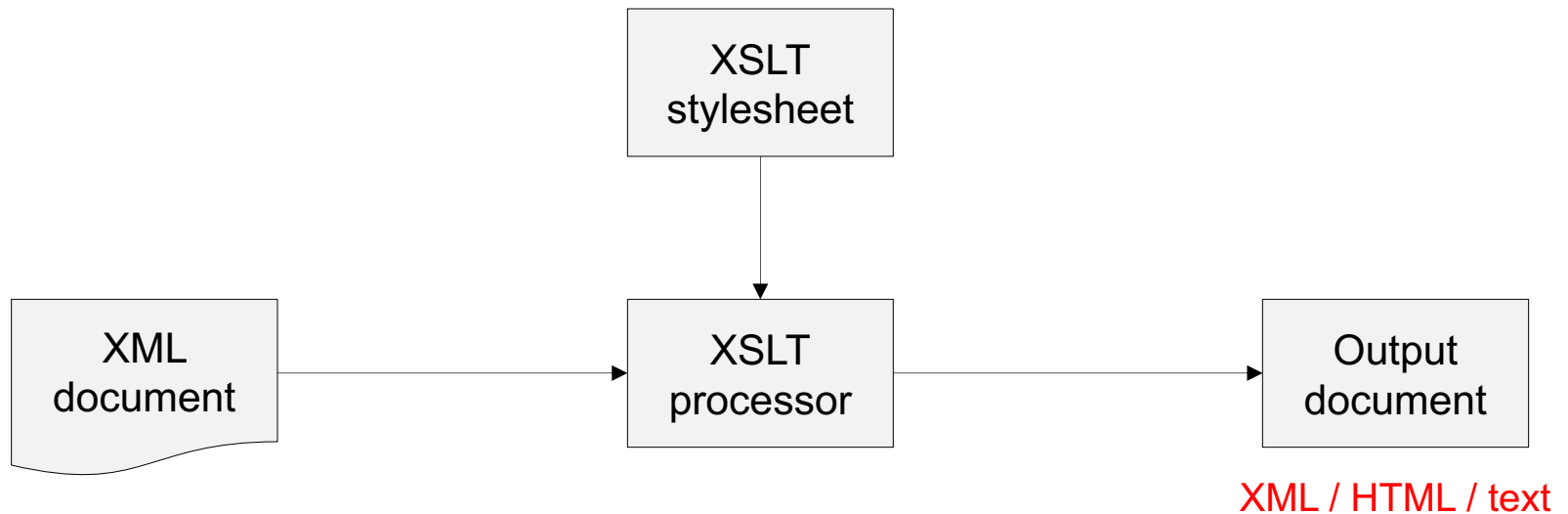
- What is XSLT?
- XSLT at First Glance
- XSLT Templates
- Creating Output
- Further Features

What is XSLT?

- XSL = eXtensible Stylesheet Language Family
- XSL = tools for styling XML documents (as CSS for HTML)
- XSLT = XSL Transformations
- XSLT is used to transform a source XML document into a target XML/HTML/text document
- XSLT uses XPath for navigation
- XSLT is a W3C standard



How XSLT Works?



- Define a transformation with an XSLT document (which is an XML document)
- Apply this transformation on an input document using an XSLT processor

XSLT at First Glance

```
<courses>
  <course semester="Summer">
    <title> SSD </title>
    <day> Thursday </day>
    <time> 09:15 </time>
    <location> HS8 </location>
  </course>
  <course semester="Winter">
    <title> Databases </title>
    <day> Tuesday </day>
    <time> 09:15 </time>
    <location> HS8 </location>
  </course>
</courses>
```

```
<html>
  <head>
    <title>Lectures Overview</title>
  </head>
  <body>
    <h1>DBAI Lectures</h1>
    <table>
      <tr><th>Semester</th><th>Title</th>
        <th>Date / Time</th><th>Location</th></tr>
      <tr><td>Summer</td><td>SSD</td>
        <td>Thursday, 09:15</td><td>HS8</td></tr>
      <tr><td>Winter</td><td>Databases</td>
        <td>Thursday, 09:15</td><td>HS8</td></tr>
    </table>
  </body>
</html>
```

XSLT at First Glance

```
<courses>
```

```
  <course semester="Summer">
```

```
    <title> SSD </title>
```

```
    <day> Thursday </day>
```

```
    <time> 09:15 </time>
```

```
    <location> HS8 </location>
```

```
  </course>
```

```
  <course semester="Winter">
```

```
    <title> Databases </title>
```

```
    <day> Tuesday </day>
```

```
    <time> 09:15 </time>
```

```
    <location> HS8 </location>
```

```
  </course>
```

```
</courses>
```

```
<html>
```

```
  <head>
```

```
    <title>Lectures Overview</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1>DBAI Lectures</h1>
```

```
    <table>
```

```
      <tr><th>Semester</th><th>Title</th>
```

```
        <th>Date / Time</th><th>Location</th></tr>
```

```
      <tr><td>Summer</td><td>SSD</td>
```

```
        <td>Thursday, 09:15</td><td>HS8</td></tr>
```

```
      <tr><td>Winter</td><td>Databases</td>
```

```
        <td>Thursday, 09:15</td><td>HS8</td></tr>
```

```
    </table>
```

```
  </body>
```

```
</html>
```

XSLT at First Glance

```
<courses>
```

```
<course semester="Summer">
  <title> SSD </title>
  <day> Thursday </day>
  <time> 09:15 </time>
  <location> HS8 </location>
</course>
```

```
<course semester="Winter">
  <title> Databases </title>
  <day> Tuesday </day>
  <time> 09:15 </time>
  <location> HS8 </location>
</course>
```

```
</courses>
```

```
<html>
```

```
<head>
```

```
<title>Lectures Overview</title>
```

```
</head>
```

```
<body>
```

```
<h1>DBAI Lectures</h1>
```

```
<table>
```

```
<tr><th>Semester</th><th>Title</th>
```

```
<th>Date / Time</th><th>Location</th></tr>
```

```
<tr><td>Summer</td><td>SSD</td>
```

```
<td>Thursday, 09:15</td><td>HS8</td></tr>
```

```
<tr><td>Winter</td><td>Databases</td>
```

```
<td>Thursday, 09:15</td><td>HS8</td></tr>
```

```
</table>
```

```
</body>
```

```
</html>
```

XSLT at First Glance

```
<courses>
  <course semester="Summer">
    <title> SSD </title>
    <day> Thursday </day>
    <time> 09:15 </time>
    <location> HS8 </location>
  </course>
  <course semester="Winter">
    <title> Databases </title>
    <day> Tuesday </day>
    <time> 09:15 </time>
    <location> HS8 </location>
  </course>
</courses>
```

```
<html>
  <head>
    <title>Lectures Overview</title>
  </head>
  <body>
    <h1>DBAI Lectures</h1>
    <table>
      <tr><th>Semester</th><th>Title</th>
        <th>Date / Time</th><th>Location</th></tr>
      <tr><td>Summer</td><td>SSD</td>
        <td>Thursday, 09:15</td><td>HS8</td></tr>
      <tr><td>Winter</td><td>Databases</td>
        <td>Thursday, 09:15</td><td>HS8</td></tr>
    </table>
  </body>
</html>
```


XSLT at First Glance

Basic principle: templates match the input document, and define the output

```
<xsl:template match="courses">
  <html>
    <head>
      <title>Lectures Overview</title>
    </head>
    <body>
      <h1>DBAI Lectures</h1>
      <table>
        <tr><th>Semester</th><th>Title</th>
          <th>Date / Time</th><th>Location</th></tr>
        <xsl:apply-templates select="course"/>
      </table>
    </body>
  </html>
</xsl:template>
```

XSLT at First Glance

Basic principle: templates match the input document, and define the output

```
<xsl:template match="course">
  <tr><td><xsl:value-of select="@semester"/></td>
    <td><xsl:value-of select="title"/></td>
    <td><xsl:value-of select="day"/>, <xsl:value-of select="time"/></td>
    <td><xsl:value-of select="location"/></td>
  </tr>
</xsl:template>
```

XSLT Documents

```
<?xml version="1.0"?>
```

```
<xsl:stylesheet version="1.0"
```

```
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```
  <xsl:output method="html"/>
```

```
  <xsl:template match="...">
```

```
    ...
```

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

```
  <xsl:template match="...">
```

```
    ...
```

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

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

XSLT documents

are

XML documents

Up to Now

- What is XSLT?
- XSLT at First Glance
- XSLT Templates
- Creating Output
- Further Features

XSLT Templates

- A template matches an element node

`<xsl:template match="pattern">`

e.g., `<xsl:template match="*">`

- A template is applied if there is a match

- Does not match child nodes automatically

`<xsl:apply-templates select="node-set-expr"/>`

Template Matching

If a **template matches** an element

- the template is executed
- by default, no templates for the subtree are called, except when explicitly applied (<xsl:apply-templates>)

```
<xsl:template match="person">
```

```
    Hello!!!
```

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

```
<xsl:template match="name">
```

```
    Name processed!!!
```

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

```
<person>
```

```
    <name> Mantas Šimkus </name>
```

```
    <email> simkus@dbai.tuwien.ac.at </email>
```

```
</person>
```

Hello!!!

xsl:apply-templates

```
<xsl:template match="person">
```

```
  Hello!!!
```

```
    <xsl:apply-templates select="name"/>
```

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

```
<xsl:template match="name">
```

```
  <xsl:value-of select="."/>
```

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

Hello!!!

Mantas Šimkus

```
<person>
```

```
  <name> Mantas Šimkus </name>
```

```
  <email> simkus@dbai.tuwien.ac.at </email>
```

```
</person>
```

xsl:apply-templates

```
<xsl:template match="person">  
    Hello!!!  
    <xsl:apply-templates select="*" />  
</xsl:template>
```

```
<xsl:template match="name | email">  
    <xsl:value-of select="." />  
</xsl:template>
```

Hello!!!

Mantas Šimkus

simkus@dbai.tuwien.ac.at

```
<person>  
    <name> Mantas Šimkus </name>  
    <email> simkus@dbai.tuwien.ac.at </email>  
</person>
```


Default Templates

- XSLT defines **default templates** that are always present
- Default templates are as follows
 - For root and elements: **apply templates for child elements**
 - For text elements: **copy content to the output**
 - For attributes: **copy value to the output**
- To **override** the behaviour of a default template create a template for an element

```
<xsl:template match=" * | / ">  
  <xsl:apply-templates select="*" />  
</xsl:template>
```

```
<xsl:template match=" text( ) | @* ">  
  <xsl:value-of select="." />  
</xsl:template>
```

Default Templates

```
<xsl:template match="person">  
    Hello!!!  
    <xsl:apply-templates select="*" />  
</xsl:template>
```

```
<xsl:template match="name">  
    <xsl:value-of select="." />  
</xsl:template>
```

Hello!!!

Mantas Šimkus

simkus@dbai.tuwien.ac.at

```
<person>  
    <name> Mantas Šimkus </name>  
    <email> simkus@dbai.tuwien.ac.at </email>  
</person>
```

Priorities

- Exactly one template is executed
- In case of more than one templates, a priority value decides which template is executed
- The XPath expression in the match attribute indicates the priority
- More specific XPath expressions have higher priority

Priorities

```
<xsl:template match="person">
```

```
  Hello!!!
```

```
  <xsl:apply-templates select="*" />
```

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

```
<xsl:template match="name">
```

```
  <xsl:value-of select="." />
```

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

```
<xsl:template match="*">
```

```
  Generic match happened!
```

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

Hello!!!

Mantas Šimkus

Generic match happened!

```
<person>
```

```
  <name> Mantas Šimkus </name>
```

```
  <email> simkus@dbai.tuwien.ac.at </email>
```

```
</person>
```

Priorities

```
<xsl:template match="person">  
    Hello!!!  
    <xsl:apply-templates select="*" />  
</xsl:template>
```

Ambiguous rule match

```
<xsl:template match="name">  
    <xsl:value-of select="." />  
</xsl:template>
```

```
<xsl:template match="name">  
    Some text!!!  
</xsl:template>
```

```
<person>  
    <name> Mantas Šimkus </name>  
    <email> simkus@dbai.tuwien.ac.at </email>  
</person>
```

Any Problem?

```
<xsl:template match="person">
  <xsl:apply-templates select="name"/>
  <xsl:apply-templates select="email"/>
</xsl:template>
```

```
<xsl:template match="name">
  Hello world!
  <xsl:value-of select="."/>
</xsl:template>
```

```
<xsl:template match="email">
  Hello world!
  <xsl:value-of select="."/>
</xsl:template>
```



same content
(bad style)

Named Templates

```
<xsl:template match="person">  
  <xsl:apply-templates select="name"/>  
  <xsl:apply-templates select="email"/>  
</xsl:template>
```

```
<xsl:template match="name | email">  
  <xsl:call-template name="output"/>  
</xsl:template>
```

```
<xsl:template name="output">  
  Here it goes this  
  <xsl:value-of select="."/>  
</xsl:template>
```

Outline

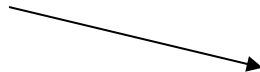
- What is XSLT?
- XSLT at First Glance
- XSLT Templates
- Creating Output
- Further Features

Creating Output

- `xsl:element`
- `xsl:attribute`
- `xsl:text`
- `xsl:comment`

Creating Output - xsl:element

```
<courses>
  <course>
    <title>SSD</title>
    <day> Thursday </day>
    <time> 09:15 </time>
  </course>
  <course>
    <title>Databases</title>
    <day> Tuesday </day>
    <time> 09:15 </time>
  </course>
</courses>
```



```
<courses>
  <SSD>
    <day> Thursday </day>
    <time> 09:15 </time>
  </SSD>
  <Databases>
    <day> Tuesday </day>
    <time> 09:15 </time>
  </Databases>
</courses>
```

Creating Output - xsl:element

```
<xsl:template match="courses">
```

```
  <courses>
```

```
    <xsl:apply-templates select="course"/>
```

```
  </courses>
```

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

```
<xsl:template match="course">
```

```
  <xsl:element name="{title/text()}">
```

```
    <day> <xsl:value-of select="day"/> </day>
```

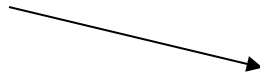
```
    <time> <xsl:value-of select="time"/> </time>
```

```
  </xsl:element>
```

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

Creating Output - xsl:attribute

```
<courses>
  <course>
    <title>SSD</title>
    <day> Thursday </day>
    <time> 09:15 </time>
  </course>
  <course>
    <title>Databases</title>
    <day> Tuesday </day>
    <time> 09:15 </time>
  </course>
</courses>
```



```
<courses>
  <SSD day="Thursday">
    <time> 09:15 </time>
  </SSD>
  <Databases day="Tuesday">
    <time> 09:15 </time>
  </Databases>
</courses>
```

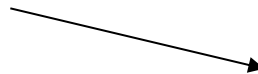
Creating Output - xsl:attribute

```
<xsl:template match="courses">  
  <courses>  
    <xsl:apply-templates select="course"/>  
  </courses>  
</xsl:template>
```

```
<xsl:template match="course">  
  <xsl:element name="{title/text()}">  
    <xsl:attribute name="day">  
      <xsl:value-of select="day"/>  
    </xsl:attribute>  
    <time> <xsl:value-of select="time"/> </time>  
  </xsl:element>  
</xsl:template>
```

Creating Output - xsl:text

```
<courses>
  <course>
    <title>SSD</title>
    <day> Thursday </day>
    <time> 09:15 </time>
  </course>
  <course>
    <title>Databases</title>
    <day> Tuesday </day>
    <time> 09:15 </time>
  </course>
</courses>
```



```
<courses>
  <SSD day="Thursday">
    Starts at
    09:15
  </SSD>
  <Databases day="Tuesday">
    Starts at
    09:15
  </Databases>
</courses>
```

Creating Output - xsl:text

```
<xsl:template match="courses">
  <courses>
    <xsl:apply-templates select="course"/>
  </courses>
</xsl:template>
<xsl:template match="course">
  <xsl:element name="{title/text()}">
    <xsl:attribute name="day">
      <xsl:value-of select="day"/>
    </xsl:attribute>
    <xsl:text> Starts at
    </xsl:text>
    <xsl:value-of select="time"/>
  </xsl:element>
</xsl:template>
```

Creating Output - xsl:comment

```
<courses>
  <course>
    <title> SSD </title>
    <day> Thursday </day>
    <time> 09:15 </time>
  </course>
  <course>
    <title> Databases </title>
    <day> Tuesday </day>
    <time> 09:15 </time>
  </course>
</courses>
```



```
<courses>
  <!-- Course Description -->
  <SSD day="Thu">
    Starts at 09:15
  </SSD>
  <Databases day="Tue">
    Starts at 09:15
  </Databases>
</courses>
```


Creating Output - xsl:comment

```
<xsl:template match="courses">
  <courses>
    <xsl:comment> Course Description </xsl:comment>
    <xsl:apply-templates select="course"/>
  </courses>
</xsl:template>

<xsl:template match="course">
  <xsl:element name="{title/text()}">
    <xsl:attribute name="day">
      <xsl:value-of select="day"/>
    </xsl:attribute>
    <xsl:text> Starts at </xsl:text>
    <xsl:value-of select="time"/>
  </xsl:element>
</xsl:template>
```

Outline

- What is XSLT?
- XSLT at First Glance
- XSLT Templates
- Creating Output
- **Further Features**

Further Features

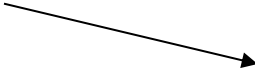
- `xsl:for-each`
- `xsl:sort`
- `xsl:if`
- `xsl:choose`
- `xsl:variable`

Further Features - xsl:for-each

```
<courses>
  <course>
    <title>SSD</title>
    <day> Thursday </day>
    <time> 09:15 </time>
  </course>
  <course>
    <title>Databases</title>
    <day> Tuesday </day>
    <time> 09:15 </time>
  </course>
</courses>
```

The structure of the course element is **not known**

We only know that the first child of each course element is title



```
<courses>
  <SSD>
    <day> Thursday </day>
    <time> 09:15 </time>
  </SSD>
  <Databases>
    <day> Tuesday </day>
    <time> 09:15 </time>
  </Databases>
</courses>
```

Further Features - xsl:for-each

```
<xsl:template match="courses">
  <courses>
    <xsl:apply-templates select="course"/>
  </courses>
</xsl:template>
```

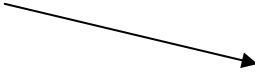
```
<xsl:template match="course">
  <xsl:element name="{title/text()}">
    <xsl:for-each select="*[position() > 1]">
      <xsl:copy-of select="."/>
    </xsl:for-each>
  </xsl:element>
</xsl:template>
```

creates a deep copy
of the selected node

Further Features - xsl:sort

```
<courses>
  <course>
    <title>SSD</title>
    <day> Thursday </day>
    <time> 09:15 </time>
  </course>
  <course>
    <title>Databases</title>
    <day> Tuesday </day>
    <time> 09:15 </time>
  </course>
</courses>
```

Sort by name



```
<courses>
  <Databases>
    <day> Tuesday </day>
    <time> 09:15 </time>
  </Databases>
  <SSD>
    <day> Thursday </day>
    <time> 09:15 </time>
  </SSD>
</courses>
```

Further Features - xsl:sort

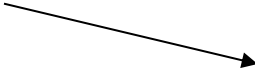
```
<xsl:template match="courses">
  <courses>
    <xsl:for-each select="course">
      <xsl:sort select="title" order="ascending"/>
      <xsl:element name="{title/text()}">
        <xsl:for-each select="*[position() > 1]">
          <xsl:copy-of select="."/>
        </xsl:for-each>
      </xsl:element>
    </xsl:for-each>
  </courses>
</xsl:template>
```

Further Features - xsl:if

```
<courses>
  <course>
    <title>SSD</title>
    <day> Thursday </day>
    <time> 09:15 </time>
  </course>
  <course>
    <day> Tuesday </day>
    <title>Databases</title>
    <time> 09:15 </time>
  </course>
</courses>
```

The structure of the course element is **not known**

The first child of each course element is **not** necessarily title



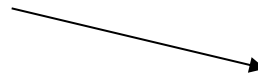
```
<courses>
  <Databases>
    <day> Tuesday </day>
    <time> 09:15 </time>
  </Databases>
  <SSD>
    <day> Thursday </day>
    <time> 09:15 </time>
  </SSD>
</courses>
```


Further Features - xsl:if

```
<xsl:template match="courses">
  <courses>
    <xsl:for-each select="course">
      <xsl:sort select="title" order="ascending"/>
      <xsl:element name="{title/text()}">
        <xsl:for-each select="*">
          <xsl:if test="local-name() != 'title' ">
            <xsl:copy-of select="."/>
          </xsl:if>
        </xsl:for-each>
      </xsl:element>
    </xsl:for-each>
  </courses>
</xsl:template>
```

Further Features - xsl:choose

```
<courses>
  <course>
    <title>SSD</title>
    <day> Thursday </day>
    <time> 09:15 </time>
  </course>
  <course>
    <title>Databases</title>
    <day> Tuesday </day>
    <time> 09:15 </time>
  </course>
</courses>
```



```
<courses>
  <Databases>
    Tuesday, 09:15.
  </Databases>
  <SSD>
    Thursday, 09:15.
  </SSD>
</courses>
```

Further Features - xsl:choose

```
<xsl:template match="courses">
  <courses>
    <xsl:for-each select="course">
      <xsl:sort select="title" order="ascending"/>
      <xsl:element name="{title/text()}">
        <xsl:for-each select="*">
          <xsl:if test="local-name() != 'title' ">
            <xsl:value-of select="."/>
            <xsl:choose>
              <xsl:when test="position() = last()">.</xsl:when>
              <xsl:otherwise>,</xsl:otherwise>
            </xsl:choose>
          </xsl:if>
        </xsl:for-each>
      </xsl:element>
    </xsl:for-each>
  </courses>
</xsl:template>
```

Further Features - xsl:variable

```
<university>
  <courses>
    <course taughtBy="L1">
      <title>SSD</title>
      <day> Thursday </day>
      <time> 09:15 </time>
    </course>
    <course taughtBy="L2">
      <title>Databases</title>
      <day> Tuesday </day>
      <time> 09:15 </time>
    </course>
  </courses>
  <lecturers>
    <lecturer id="L1">Mantas Šimkus</lecturer>
    <lecturer id="L2">Reinhard Pichler</lecturer>
  </lecturers>
</university>
```



```
<courses>
  SSD taught by Mantas Šimkus
  Databases taught by Reinhard Pichler
</courses>
```

Further Features - xsl:variable

```
<xsl:template match="university">  
  <xsl:apply-templates select="courses"/>  
</xsl:template>
```

```
<xsl:template match="courses">  
  <courses>  
    <xsl:apply-templates select="course"/>  
  </courses>  
</xsl:template>
```

```
<xsl:template match="course">  
  <xsl:value-of select="title"/> taught by <xsl:value-of select="//lecturer[@id = @taughtBy]"/>  
</xsl:template>
```

Result:

```
<courses>  
  SSD taught by  
  Databases taught by  
</courses>
```

?

Further Features - xsl:variable

```
<xsl:template match="university">  
  <xsl:apply-templates select="courses"/>  
</xsl:template>
```

```
<xsl:template match="courses">  
  <courses>  
    <xsl:apply-templates select="course"/>  
  </courses>  
</xsl:template>
```

```
<xsl:template match="course">  
  <xsl:variable name="var" select="@taughtBy"/>  
  <xsl:value-of select="title"/> taught by <xsl:value-of select="//lecturer[@id = $var]"/>  
</xsl:template>
```

Sum Up

- What is XSLT?
- XSLT at First Glance
- XSLT Templates
- Creating Output
- Further Features