4. XSLT

PB138

XSLT

- people.xml + people.xsl = xsl-result.html
- XSLT is composed of templates.

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

Running the Transformation

- Try transformation from Netbeans and from Java
- HELLO WORLD

Basic Transformations

- **/** vs **/people**
- Evaluation of the algorithm
  - apply-templates will again apply the whole template file to each child
  - text() matching
- The implicit templates
  - We can override them!

Implicit Templates

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

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

Template Priority

```
<xsl:template match="person[@name=’Filip’]"
   priority=”2”> <!-- highest priority -->
  ....
</xsl:template>

<xsl:template match="person”
   priority=”1”>
  ....
</xsl:template>
```

Processing Modes

```
<xsl:template match="person[@name=’Filip’]"
   mode=”nameBlue”>
  ....
</xsl:template>

<xsl:apply-templates mode="nameBlue"/>
```

Modes, foreach and IFs

- <xsl:choose>
- <xsl:if>
- <xsl:for-each>

Functional calling

- create a table "id, name"
  - Steven Segal in red color

Other important concepts

- xsl:sort
- xsl:element
- xsl:call-template (with parameters)

xsl:element, xsl:attribute

```
<xsl:element name="{@element}">
<xsl:attribute name="id">ID1</xsl:attribute>
```

- Output the names in "id" elements:

```
<el1>
  Filip Nguyen
</el1>
```