Introduction to XSLT

What is XSL

- XSL stands for eXtensible Stylesheet Language
- XSLT transforms an XML document into another XML document
- XSL is a XML language

What is XSL

- XSLT - a language for transforming XML documents
- XPath - a language for navigating in XML documents
- XSL-FO - a language for formatting XML documents

What is XSL

- XSLT - a language for transforming XML documents
- XPath - a language for navigating in XML documents
- XSL-FO - a language for formatting XML documents

Why XSL

- CSS adds styles (font, color) to HTML elements
- XSL = XML Style Sheets – allows you to transform an XML file into an HTML file or another text-based format

Why two style sheet languages?

	CSS	XSL
Can be used with HTML?	yes	no
Can be used with XML?	yes	yes
Transformation language?	no	yes
Syntax	CSS	XML

XSLT

Why XSLT?

Why XSLT?

- XML is not a fixed tag set (like HTML) and has no (application) semantics

Why XSLT?

- XML is not a fixed tag set (like HTML) and has no (application) semantics
- XML markup does not (usually) include formatting information

Why XSLT?

- XML is not a fixed tag set (like HTML) and has no (application) semantics
- XML markup does not (usually) include formatting information
- Reuse: the same content can look different in different contexts

Why XSLT?

- XML is not a fixed tag set (like HTML) and has no (application) semantics
- XML markup does not (usually) include formatting information
- Reuse: the same content can look different in different contexts
- Multiple output formats: different media (paper, online), different sizes (manuals, reports), different classes of output devices (workstations, hand-held devices)

Why XSLT?

- XML is not a fixed tag set (like HTML) and has no (application) semantics
- XML markup does not (usually) include formatting information
- Reuse: the same content can look different in different contexts
- Multiple output formats: different media (paper, online), different sizes (manuals, reports), different classes of output devices (workstations, hand-held devices)
- Styles tailored to the reader's preference (e.g., accessibility): print size, color, simplified layout for audio readers

Why XSLT?

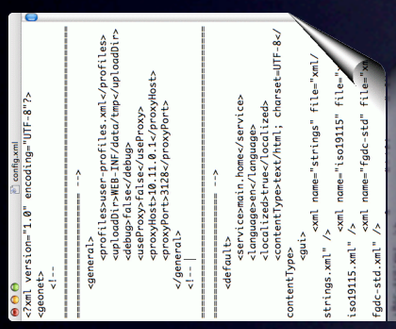
- XML is not a fixed tag set (like HTML) and has no (application) semantics
- XML markup does not (usually) include formatting information
- Reuse: the same content can look different in different contexts
- Multiple output formats: different media (paper, online), different sizes (manuals, reports), different classes of output devices (workstations, hand-held devices)
- Styles tailored to the reader's preference (e.g., accessibility): print size, color, simplified layout for audio readers
- Standardized styles: corporate stylesheets can be applied to the content at any time

Why XSLT?

- XML is not a fixed tag set (like HTML) and has no (application) semantics
- XML markup does not (usually) include formatting information
- Reuse: the same content can look different in different contexts
- Multiple output formats: different media (paper, online), different sizes (manuals, reports), different classes of output devices (workstations, hand-held devices)
- Styles tailored to the reader's preference (e.g., accessibility): print size, color, simplified layout for audio readers
- Standardized styles: corporate stylesheets can be applied to the content at any time
- Freedom from style issues for content authors: technical writers needn't be concerned with layout issues because the correct style can be applied later

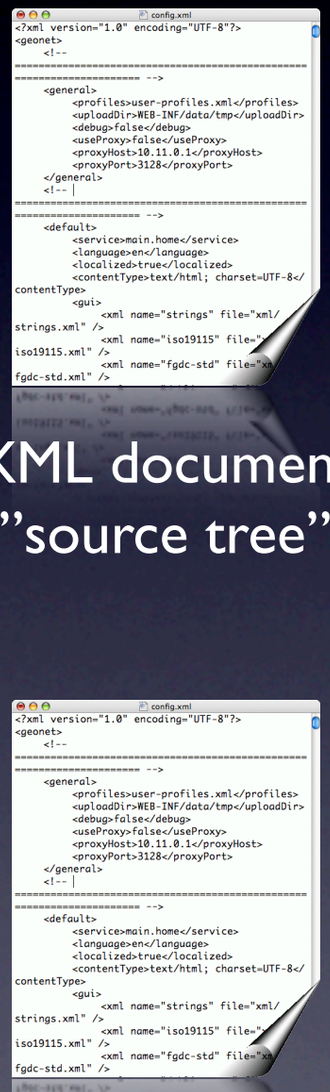
The basic idea of XSLT

The basic idea of XSLT



XML document
(”source tree”)

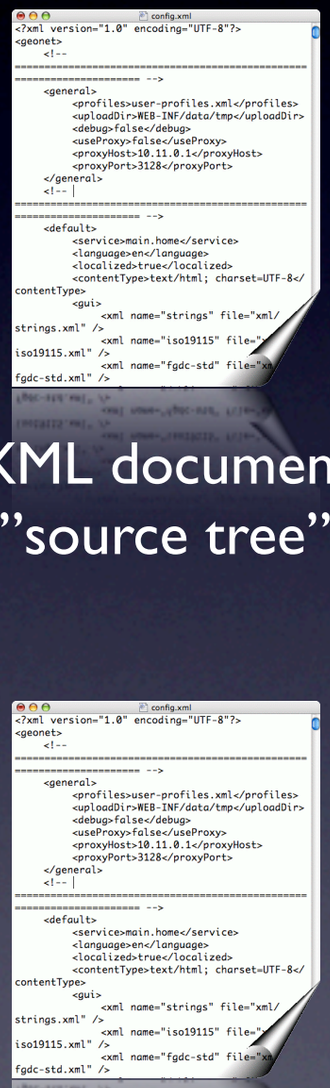
The basic idea of XSLT



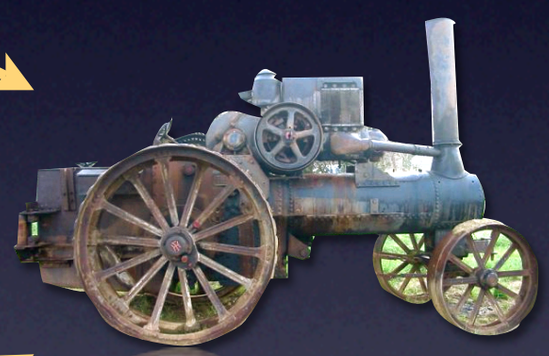
XML document
(”source tree”)

XSLT stylesheet

The basic idea of XSLT



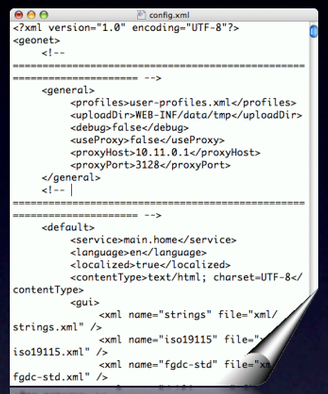
XML document
(”source tree”)



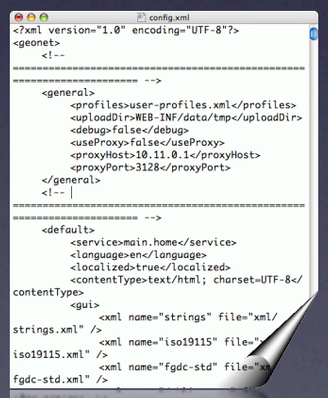
XSLT processor

XSLT stylesheet

The basic idea of XSLT

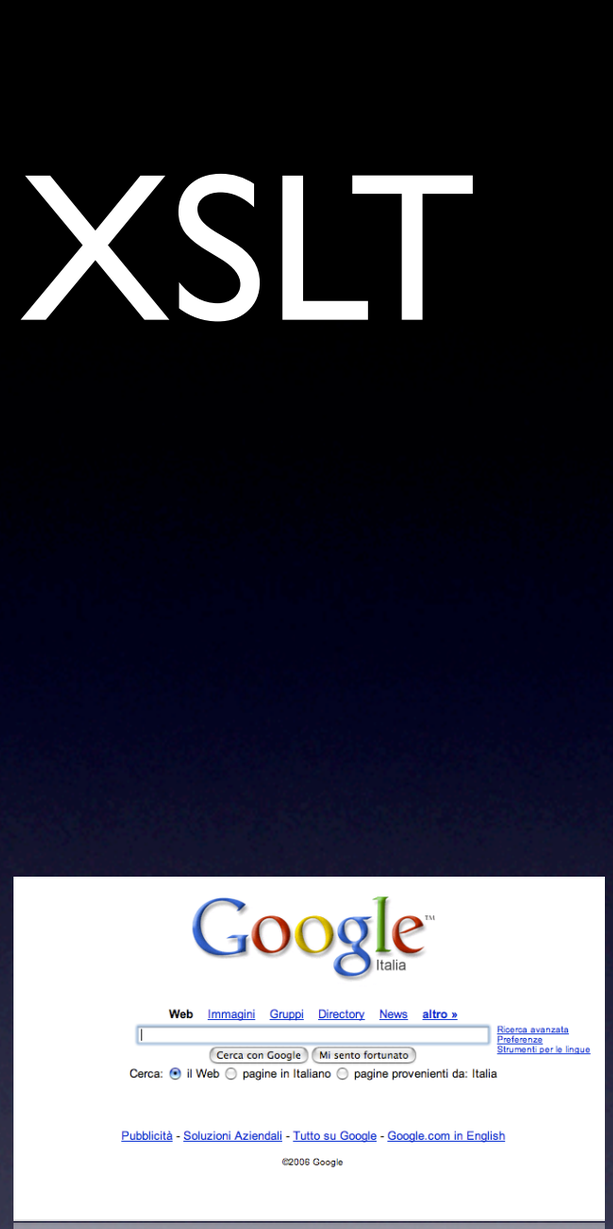


XML document
("source tree")



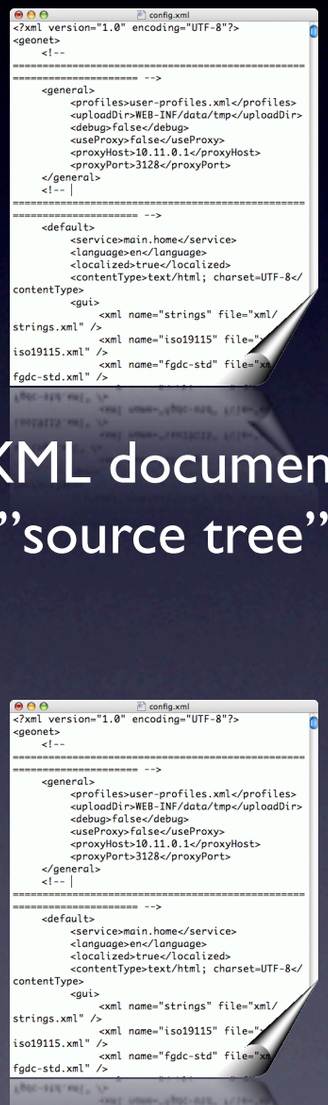
XSLT stylesheet

XSLT processor

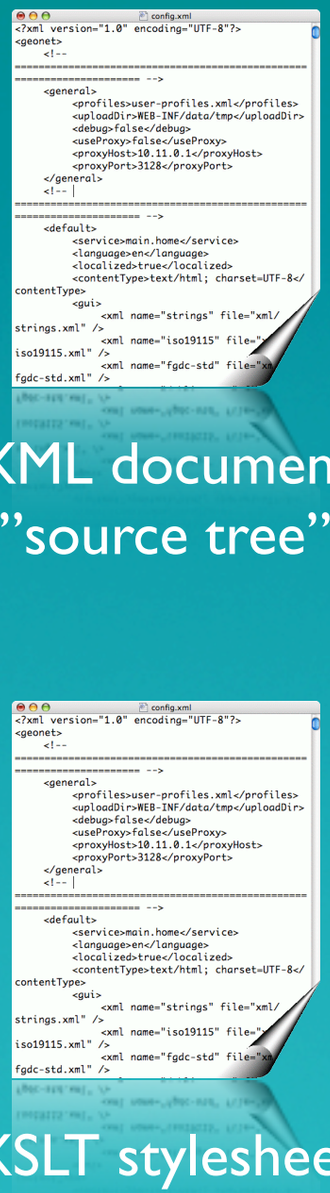


Web page

The basic idea of XSLT



The basic idea of XSLT



XML document
("source tree")

XSLT stylesheet

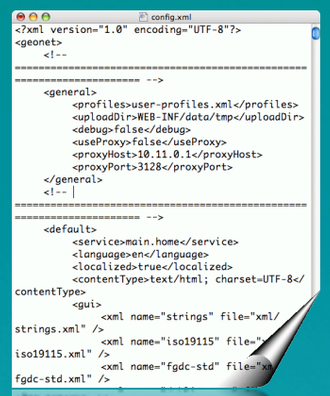
XSLT processor

Server

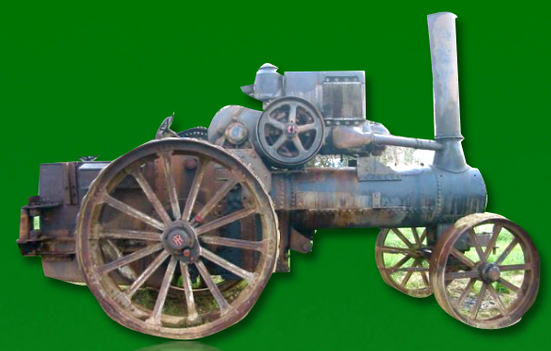
Web page

Browser

The basic idea of XSLT



XML document
("source tree")



XSLT processor

Web page

XSLT stylesheet

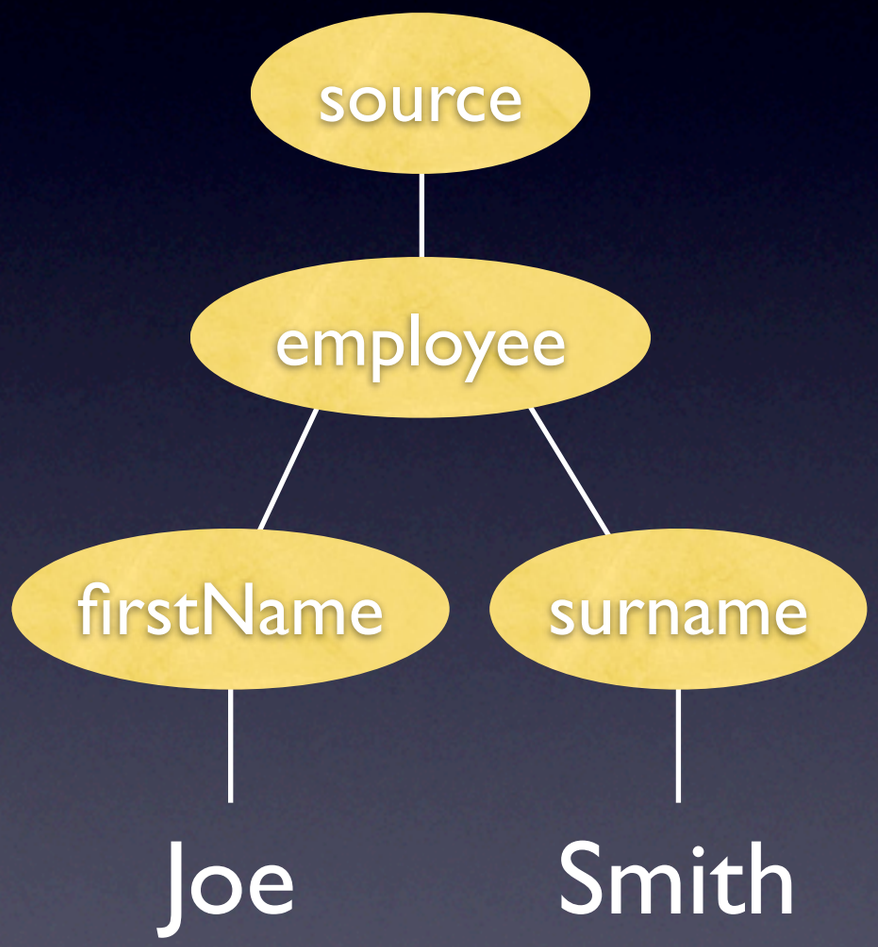
Server

Browser

Simple XSLT transformation

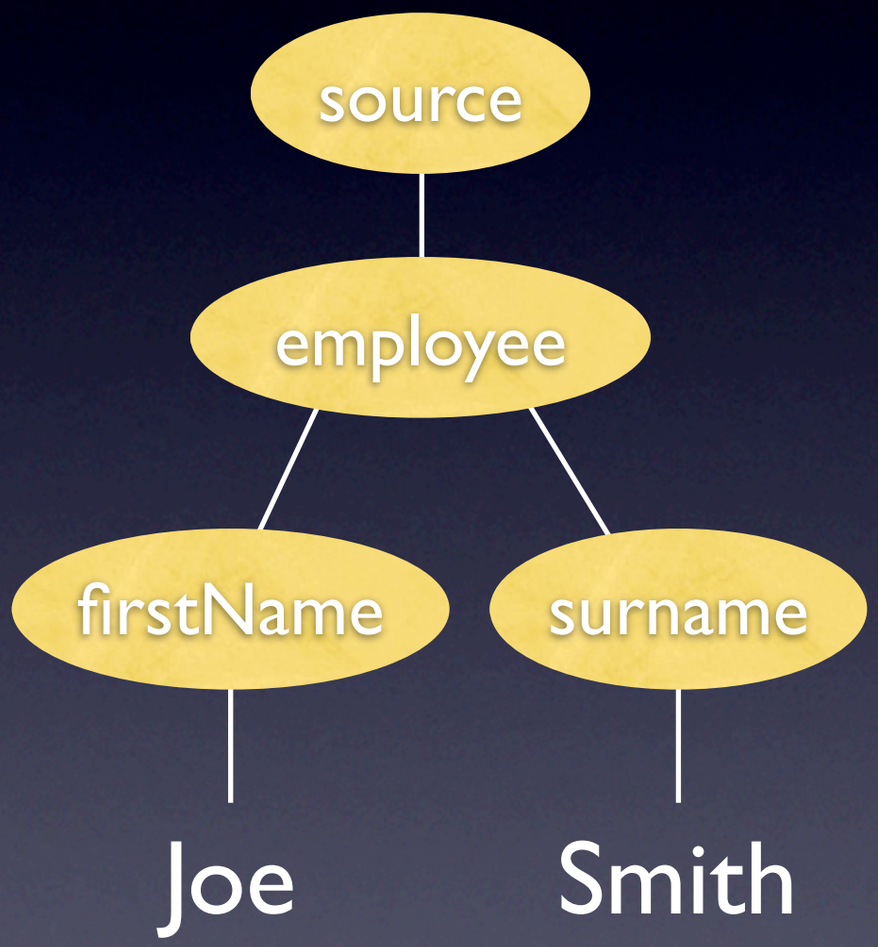
```
<source>
  <employee>
    <firstName>Joe</firstName>
    <surname>Smith</surname>
  </employee>
</source>
```

Simple XSLT transformation




```
<xsl:template match="/">
<html>
  <body>
    <h1>My employee</h1>
      <xsl:value-of select="employee/firstName"/>
      <xsl:value-of select="employee/surname"/>
  </body>
</html>
</xsl:template>
```

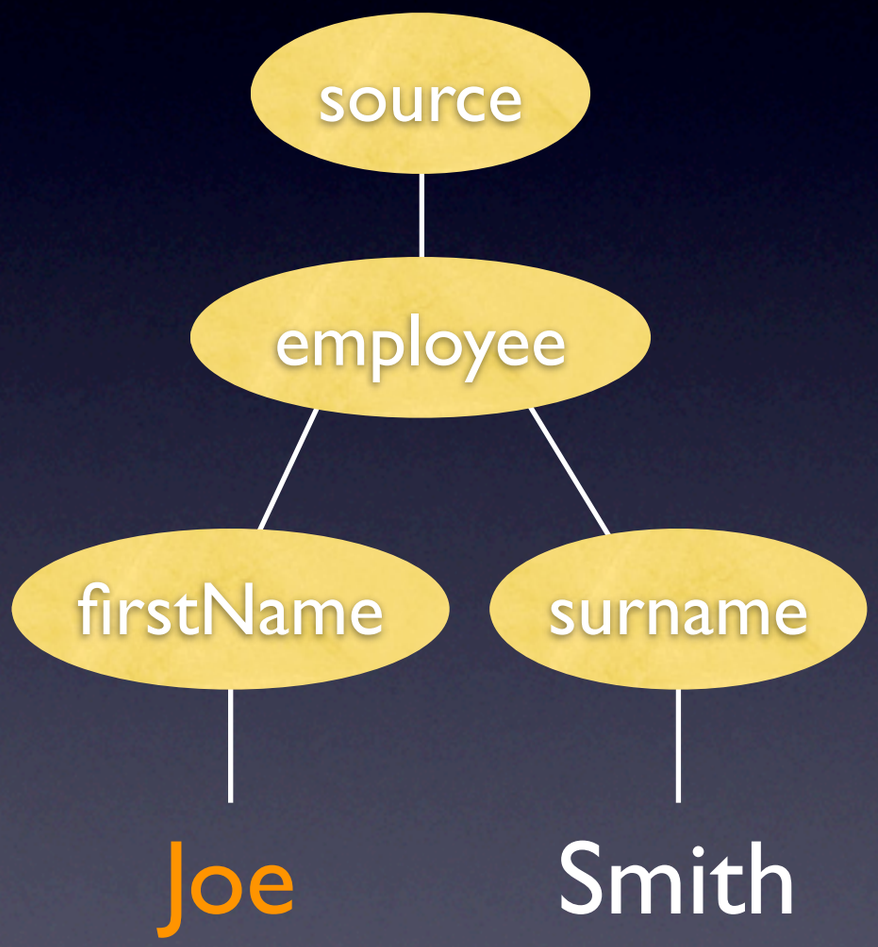
Simple XSLT transformation




```
<xsl:template match="/">
<html>
  <body>
    <h1>My employee</h1>
      <xsl:value-of select="employee/firstName"/>
      <xsl:value-of select="employee/surname"/>
  </body>
</html>
</xsl:template>
```

```
<html>
  <body>
    <h1>My employee</h1>
      <xsl:value-of select="employee/firstName"/>
      <xsl:value-of select="employee/firstName"/>
  </body>
</html>
```

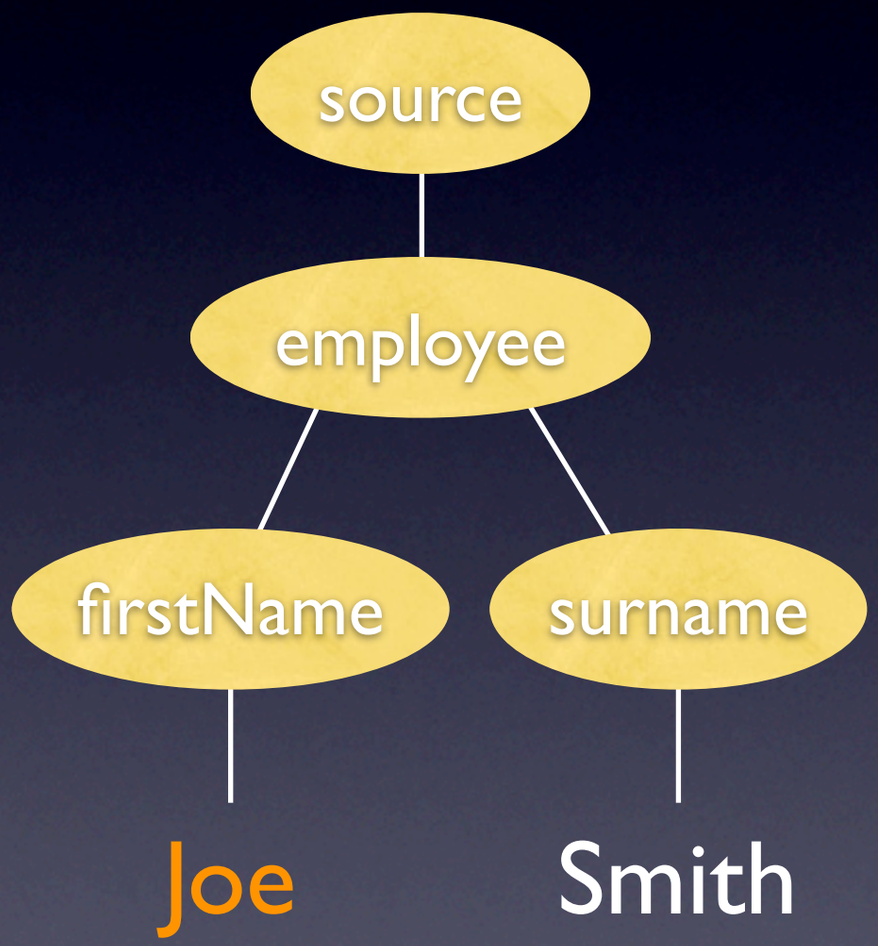
Simple XSLT transformation




```
<xsl:template match="/">
<html>
  <body>
    <h1>My employee</h1>
      <xsl:value-of select="employee/firstName"/>
      <xsl:value-of select="employee/surname"/>
  </body>
</html>
</xsl:template>
```

```
<html>
  <body>
    <h1>My employee</h1>
      <xsl:value-of select="employee/firstName"/>
      <xsl:value-of select="employee/firstName"/>
  </body>
</html>
```

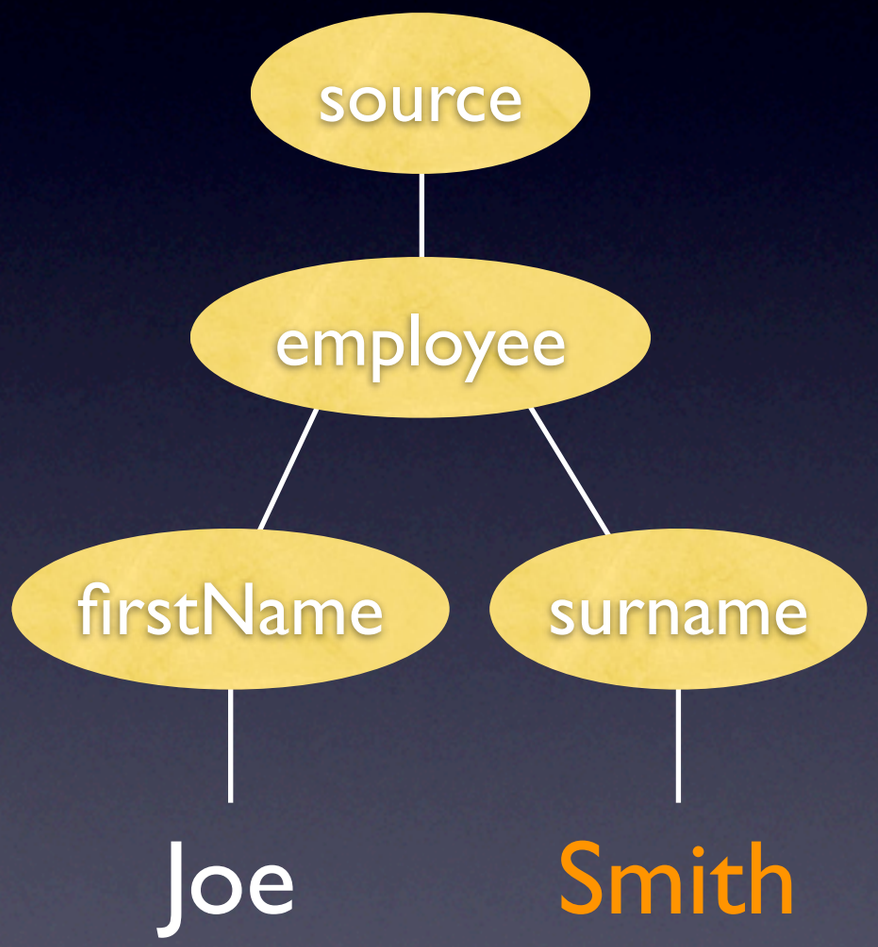
Simple XSLT transformation




```
<xsl:template match="/">
<html>
  <body>
    <h1>My employee</h1>
      <xsl:value-of select="employee/firstName"/>
      <xsl:value-of select="employee/surname"/>
  </body>
</html>
</xsl:template>
```

```
<html>
  <body>
    <h1>My employee</h1>
      Joe
      <xsl:value-of select="employee/firstName"/>
  </body>
</html>
```

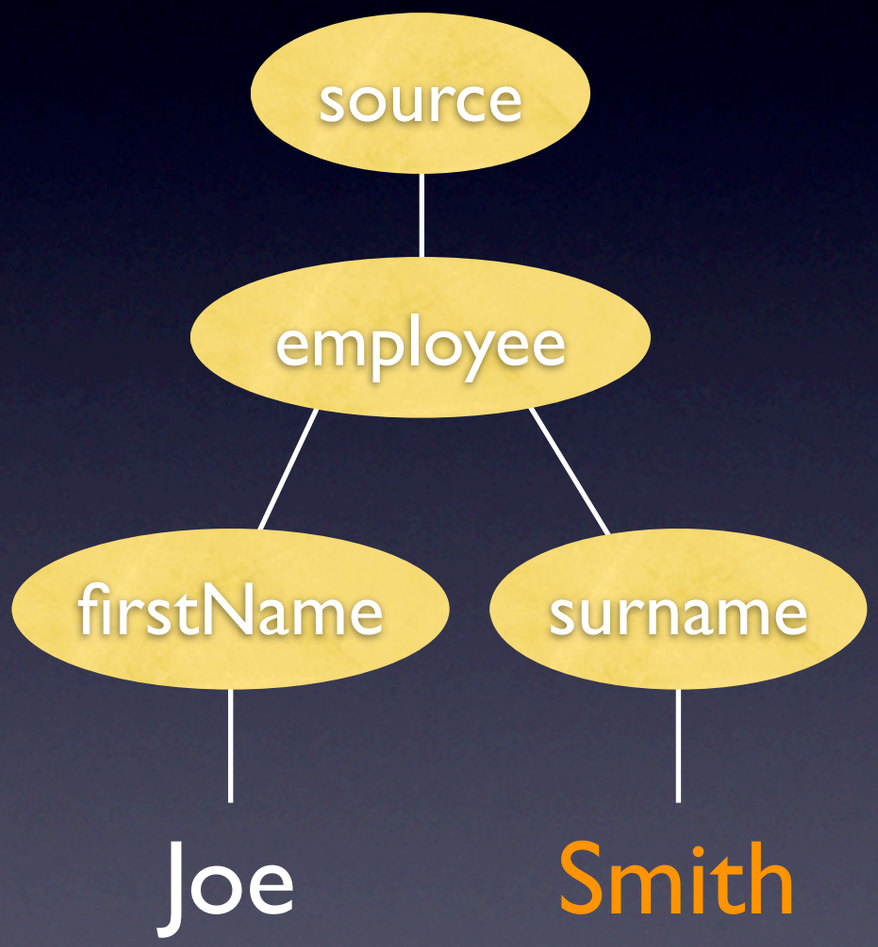
Simple XSLT transformation




```
<xsl:template match="/">
<html>
  <body>
    <h1>My employee</h1>
      <xsl:value-of select="employee/firstName"/>
      <xsl:value-of select="employee/surname"/>
  </body>
</html>
</xsl:template>
```

```
<html>
  <body>
    <h1>My employee</h1>
      Joe
      <xsl:value-of select="employee/firstName"/>
  </body>
</html>
```

Simple XSLT transformation



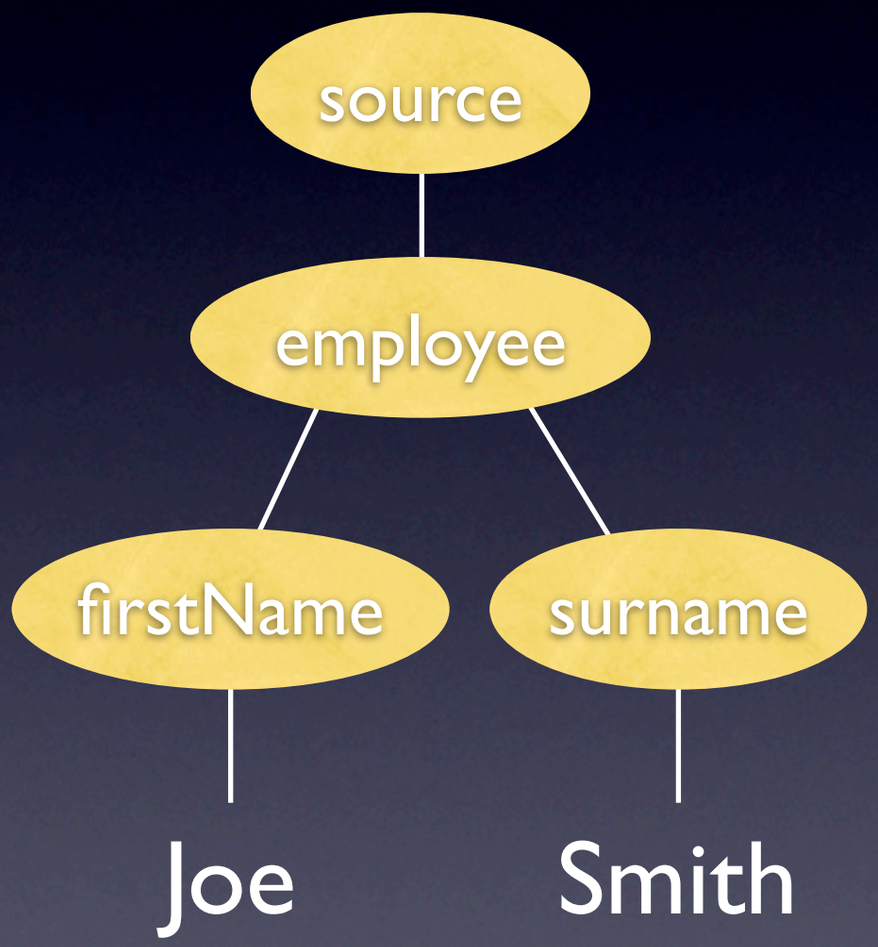

```
<xsl:template match="/">
<html>
  <body>
    <h1>My employee</h1>
      <xsl:value-of select="employee/firstName"/>
      <xsl:value-of select="employee/surname"/>
  </body>
</html>
</xsl:template>
```

```
<html>
  <body>
    <h1>My employee</h1>
      Joe
      Smith
  </body>
</html>
```

Result

```
<html>
  <body>
    <h1>My employee</h1>
      Joe
      Smith
  </body>
</html>
```

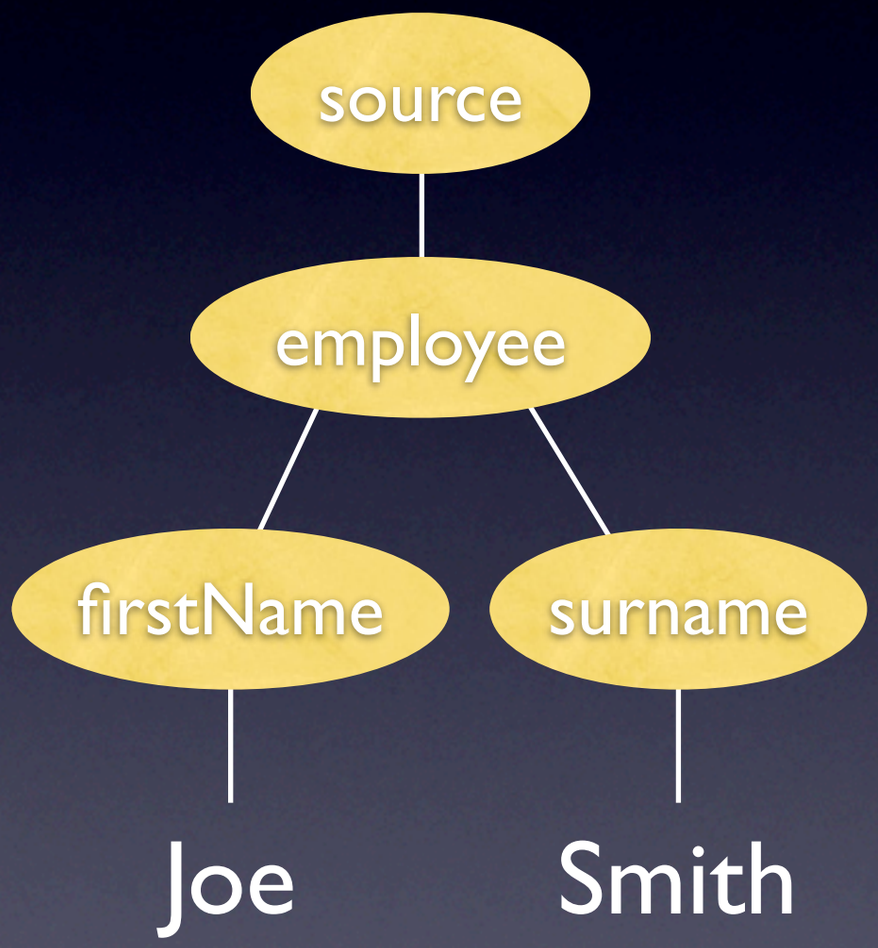
Simple XSLT transformation




```
<xsl:template match="/">
<html>
  <body>
    <h1>My employee</h1>
      <xsl:value-of select="employee/firstName"/>
      <xsl:value-of select="employee/surname"/>
  </body>
</html>
</xsl:template>
```

employee/firstName is a XPath expression

Simple XSLT transformation

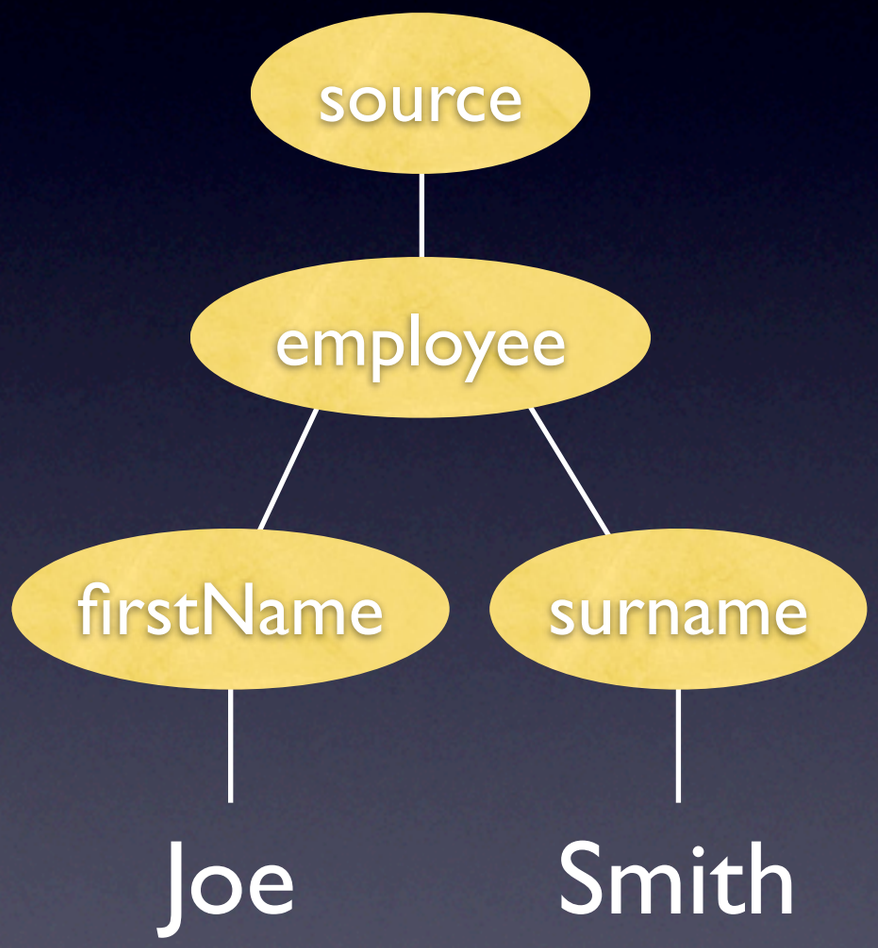



```
<xsl:template match="/">
<html>
  <body>
    <h1>My employee</h1>
      <xsl:apply-templates select=”employee/
        firstName”/>
      <xsl:apply-templates select=”employee/
        surname”/>
  </body>
</html>
</xsl:template>

<xsl:template match=”firstName”>
  <b><xsl:value-of select=”.”/></b>
</xsl:template>

<xsl:template match=”surname”>
  <i><xsl:value-of select=”.”/></i>
</xsl:template>
```

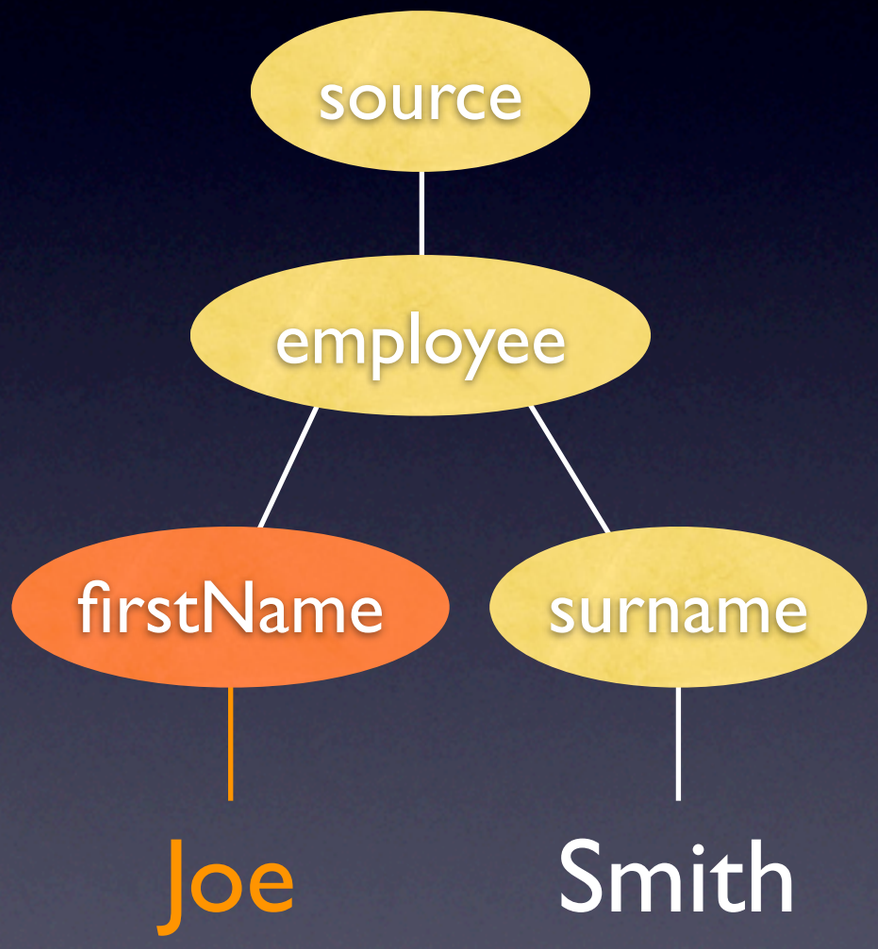
Simple XSLT transformation




```
<xsl:template match="/">
<html>
  <body>
    <h1>My employee</h1>
      <xsl:apply-templates select=”employee/
        firstName”/>
      <xsl:apply-templates select=”employee/
        surname”/>
  </body>
</html>
</xsl:template>
```

```
<html>
  <body>
    <h1>My employee</h1>
      <xsl:apply-templates select=”employee/
        firstName”/>
      <xsl:apply-templates select=”employee/
        surname”/>
  </body>
</html>
```

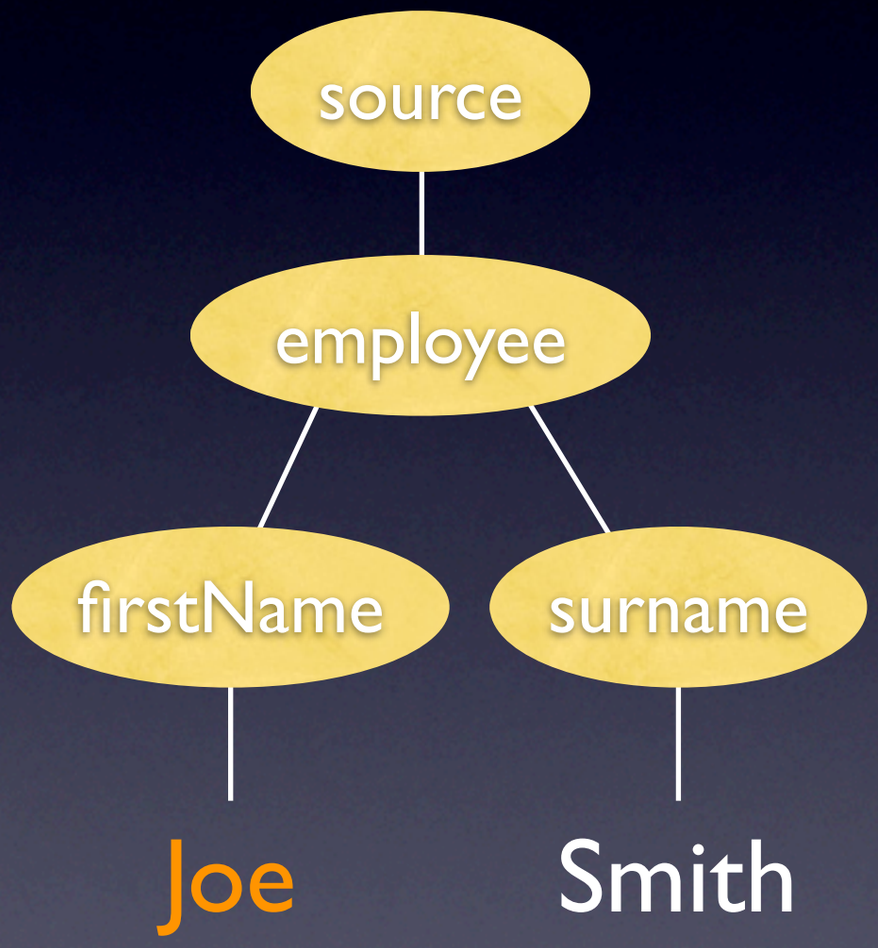
Simple XSLT transformation




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

```
<html>
  <body>
    <h1>My employee</h1>
      <xsl:apply-templates select="employee/
        firstName"/>
      <xsl:apply-templates select="employee/
        surname"/>
  </body>
</html>
```

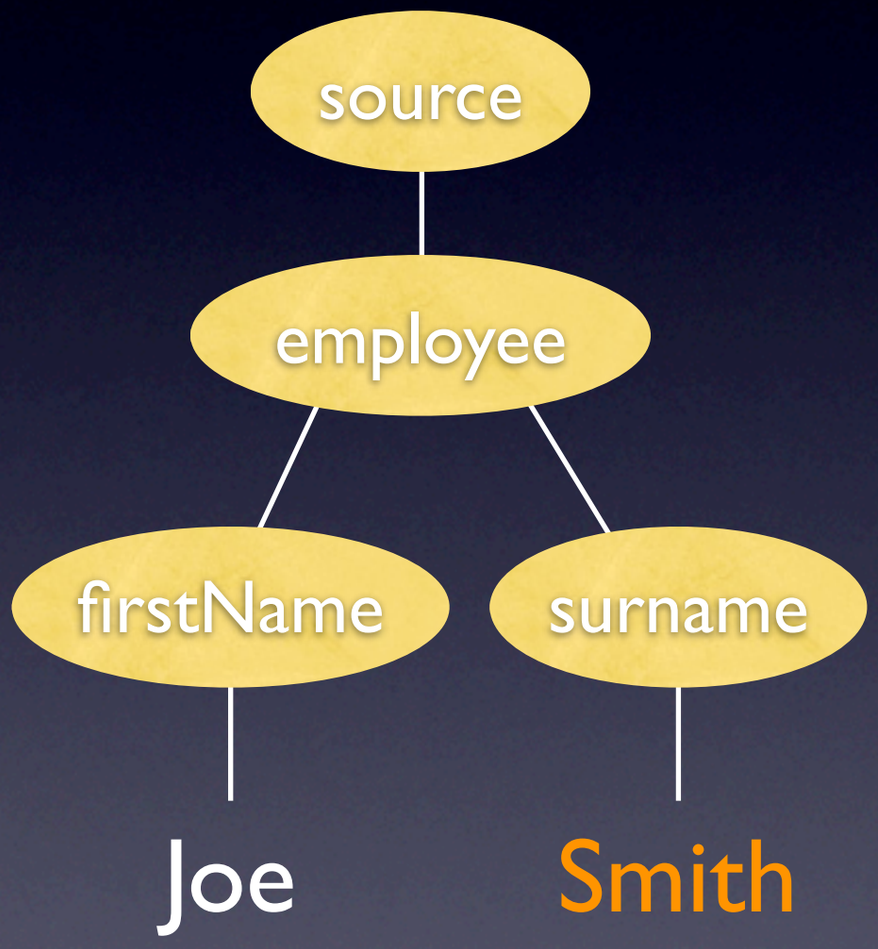
Simple XSLT transformation




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

```
<html>
  <body>
    <h1>My employee</h1>
      <b>Joe</b>
      <xsl:apply-templates select="employee/
        surname"/>
  </body>
</html>
```

Simple XSLT transformation




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

```
<html>
  <body>
    <h1>My employee</h1>
      <b>Joe</b>
      <i>Smith</i>
  </body>
</html>
```

Result

```
<html>
  <body>
    <h1>My employee</h1>
      <b>Joe</b>
      <i>Smith</i>
  </body>
</html>
```

The <xsl:for-each> Element

```
<table>
  <xsl:for-each select=”employee”>
    <tr>
      <td>
        <xsl:value-of select=”firstName”/>
      </td>
      <td>
        <xsl:value-of select=”surname”/>
      </td>
    </tr>
  </xsl:for-each>
</table>
```

Joe	Smith
Eric	Gibson
Ada	Hook

The <xsl:sort> Element

```
<table>
  <xsl:for-each select=”employee”>
    <xsl:sort select="firstName"/>
    <tr>
      <td>
        <xsl:value-of select=”firstName”/>
      </td>
      <td>
        <xsl:value-of select=”surname”/>
      </td>
    </tr>
  </xsl:for-each>
</table>
```

Ada	Hook
Joe	Smith
Eric	Gibson

The <xsl:if> Element

```
<table>
  <xsl:for-each select=”employee”>
    <xsl:if test="surname=‘Smith’">
    <tr>
      <td>
        <xsl:value-of select=”firstName”/>
      </td>
      <td>
        <xsl:value-of select=”surname”/>
      </td>
    </tr>
  </xsl:for-each>
</table>
```

Joe	Smith

XPath

XPath

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```

Selecting nodes

Expression	Description
nodename	Selects all child nodes of the node
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

Selecting nodes

/bookstore/book

Selects all attributes that are named lang

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```

Selecting nodes

//@lang

Selects all attributes that are named lang

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```

Predicates

/bookstore/book[1]

Selects the first book element that is the child of the bookstore element

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```

Predicates

/bookstore/book[price>35.00]

Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```

References

- W3C
  - http://www.w3.org/Style/XSL/
- Tutorials
  - http://www.w3schools.com/xsl/
  - http://www.topxml.com/xsl/tutorials/intro/default.asp