

# Generating Web Pages Using XSLT

# XSLT for Data Interchange

# 6.1.xml: An Employee List Document

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<employeelist>
```

```
  <!-- This is just a copy of ex_3.7.xml with this  
        comment changed -->
```

```
<listdate>April 4, 2006</listdate>
```

```
<listregion>Central</listregion>
```

```
<employee>
```

```
  <name>
```

```
    <firstname>Xavier</firstname>
```

```
    <lastname>Ampul</lastname>
```

```
    <suffix>III</suffix>
```

```
  </name>
```

```
  <address>
```

```
    <street>1234 Main Street</street>
```

```
    <street>Apartment 401</street>
```

```
    <city>Chicago</city>
```

```
    <state>IL</state>
```

## 6.1.xml: An Employee List Document (cont.)

```
<zip>60610</zip>
</address>
<phone type="home">555-555-5555</phone>
</employee>

<employee>
  <name>
    <firstname>Frances</firstname>
    <middleinit>R</middleinit>
    <lastname>Smith</lastname>
  </name>
  <address>
    <street>559 Primary Avenue</street>
    <city>Evanston</city>
    <state>IL</state>
    <zip>60201</zip>
  </address>
  <phone type="home">555-555-5590</phone>
  <phone type="mobile">555-555-5591</phone>
</employee>
```

# 6.1.xml: An Employee List Document (last)

```
<employee>
  <name>
    <firstname>Raymond</firstname>
    <lastname>Jones</lastname>
  </name>
  <address>
    <street>987 Center Street</street>
    <city>Chicago</city>
    <state>IL</state>
    <zip>60610</zip>
  </address>
  <phone type="mobile">555-555-0987</phone>
</employee>

</employeelist
```

# 6.1.xslt: An Empty XSLT Stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text"/>

  <!-- This is an empty stylesheet.  -->

</xsl:stylesheet>
```

# 6.1.txt: Results from an Empty Stylesheet

```
April 4, 2006CentralXavierAmpuIIII1234 Main StreetApartment 401ChicagoIL
60610555-555-5555FrancesRSmith559 Primary AvenueEvanstonIL6020155
5-555-5590555-555-5591RaymondJones987 Center StreetChicagoIL60610
555-555-0987
```

# About Example 6.1

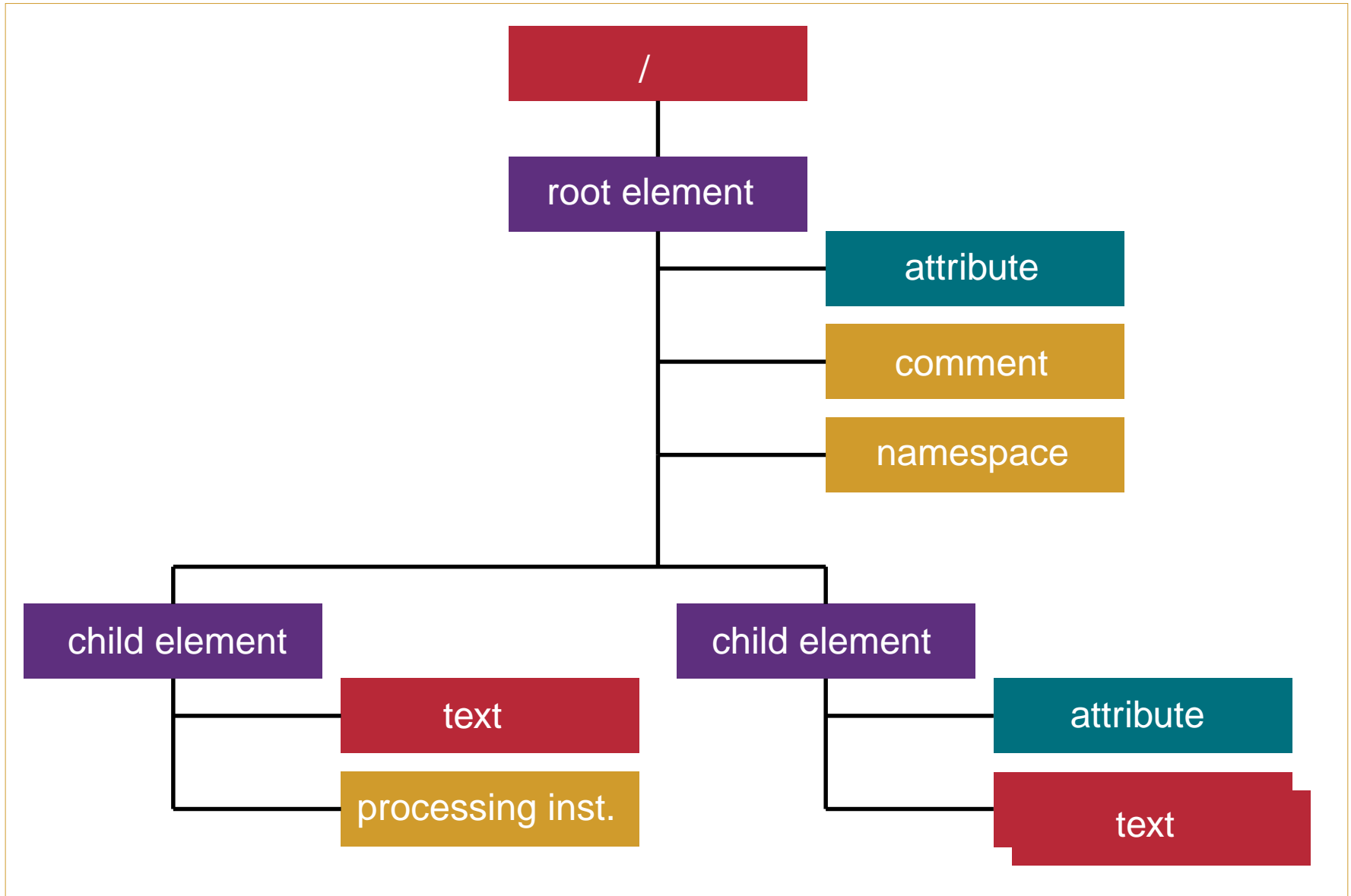
- XSLT Stylesheets have default behavior.
- So, an empty stylesheet (one without **template rules**) creates output.
- The default navigation behavior is for the processor to visit each of the **element nodes** in the **document tree**.
- At each element node, the default behavior is to output the text value of the element node.
- The root element of an XSLT stylesheet can be `xsl:stylesheet` or `xsl:transform`.
- `xsl:stylesheet` is most often used in practice.
- The `xsl:output` element controls output options.



# About Example 6.1 (last)

- In this example, the `xs1:output` element causes a normal text file to be created rather than an XML or HTML document.

# The XSLT Tree Model



# 6.2.xslt: Hello World

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xhtml"
    doctype-system=
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
    encoding="UTF-8" indent="yes"/>

  <xsl:template match="/">
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
        <title>Hello World</title>
        <meta http-equiv="Content-Type" content="text/html; charset=U
      </head>
      <body>
        <h2>Hello, World!</h2>
      </body>
    </html>
  </xsl:template>
```

## 6.2.xslt: Hello World (last)

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

---

# Hello, World!

# About Example 6.2

- This stylesheet is modeled after the **fill-in-the-blanks** stylesheet pattern.
- A single **template rule** is created with the `xsl:template` element.
- This single rule matches the root of the document tree.
- When a node from the document tree matches a template rule, the rule fires and its template text is placed into the output tree.
- Also, the automatic navigation to the node's children is interrupted.

## About Example 6.2 (last)

- All template text is inserted into the output tree via this single rule.
- In this case, we haven't even filled in the blanks from the input document.
- Our output is all fixed literal text.
- Notice that we have included the basic markup that forms a proper XHTML page.
- The attribute values on the `xsl:output` element cause an html document to be created with the proper `DOCTYPE` declaration, `UTF-8` encoding, and useful indentation.

## 6.3.xslt: Starting the Employee List Page

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xhtml"
    doctype-system=
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
    encoding="UTF-8" indent="yes"/>

  <xsl:template match="/">
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
        <title>Employee List</title>
      </head>
      <body>
        <h2>Xampul Employee List</h2>
        <h3><xsl:value-of select="/employee/listregion"/> Region</h3>
        <h3>Date: <xsl:value-of select="/employee/listdate"/></h3>
      </body>
    </html>
  </xsl:template>
```



## 6.3.xslt: Starting the Employee List Page (last)

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

# **Xampul Employee List**

**Central Region**

**Date: April 4, 2006**

# About Example 6.3

- The `xsl:value-of` element is used to pull values from the input tree and place them in the output tree.
- The `select` attribute takes an **XPATH** expression which gives an absolute path to the element that we desire.
- The value returned is the text that is contained in the element that we have selected.
- We have used this fill-in-the-blanks approach to create the headings of our page.

## 6.4.xslt: Listing the Names

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xhtml"
    doctype-system=
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
    encoding="UTF-8" indent="yes"/>

  <xsl:template match="/">
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
        <title>Employee List</title>
      </head>
      <body>
        <h2>Xampul Employee List</h2>
        <h3><xsl:value-of select="/employee/listregion"/> Region</h3>
        <h3>Date: <xsl:value-of select="/employee/listdate"/></h3>
        <xsl:for-each select="employeelist/employee">
          <p>
            <xsl:value-of select="name/firstname"/>
```

## 6.4.xslt: Listing the Names (last)

```
<xsl:text> </xsl:text>
<xsl:value-of select="name/middleinit"/>
<xsl:text>. </xsl:text>
<xsl:value-of select="name/lastname"/>
<xsl:text> </xsl:text>
<xsl:value-of select="name/suffix"/>
  </p>
</xsl:for-each>
</body>
</html>
</xsl:template>

</xsl:stylesheet>
```

# **Xampul Employee List**

**Central Region**

**Date: April 4, 2006**

Xavier . Ampul III

Frances R. Smith

Raymond . Jones

# About Example 6.4

- The new code in this example follows the **navigational** design pattern for stylesheets.
- `xsl:for-each` is used to iterate over each of a repeating group of elements.
- Within the `xsl:for-each` element, a **context** is set.
- So, all XPATH expressions within the loop construct are made relative to the established context.
- A series of `xsl:value-of` elements are used within the loop to pull out the values for the name.
- `xsl:text` is a reliable way to force text into the output tree (especially spaces).

# About Example 6.4 (last)

- Notice that we need to make adjustments to the code in order to properly treat optional elements.



## 6.5.xslt: Fix Name and Do Address

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xhtml"
    doctype-system=
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
    encoding="UTF-8" indent="yes"/>

  <xsl:template match="/">
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
        <title>Employee List</title>
      </head>
      <body>
        <h2>Xampul Employee List</h2>
        <h3><xsl:value-of select="/employee/listregion"/> Region</h3>
        <h3>Date: <xsl:value-of select="/employee/listdate"/></h3>
        <xsl:for-each select="employeelist/employee">
          <p>
            <xsl:value-of select="name/firstname"/>
```

## 6.5.xslt: Fix Name and Do Address (cont.)

```
<xsl:text> </xsl:text>
<xsl:if test="name/middleinit">
  <xsl:value-of select="name/middleinit"/>
  <xsl:text>. </xsl:text>
</xsl:if>
<xsl:value-of select="name/lastname"/>
<xsl:if test="name/suffix">
  <xsl:text> </xsl:text>
  <xsl:value-of select="name/suffix"/>
</xsl:if>
<br/>
<xsl:for-each select="address/street">
  <xsl:value-of select="."/>
  <br/>
</xsl:for-each>
<xsl:value-of select="address/city"/>
<xsl:text>, </xsl:text>
<xsl:value-of select="address/state"/>
<xsl:text> </xsl:text>
<xsl:value-of select="address/zip"/>
```

## 6.5.xslt: Fix Name and Do Address (last)

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

</xsl:stylesheet>
```

# Xampul Employee List

## Central Region

**Date: April 4, 2006**

Xavier Ampul III  
1234 Main Street  
Apartment 401  
Chicago, IL 60610

Frances R. Smith  
559 Primary Avenue  
Evanston, IL 60201

Raymond Jones  
987 Center Street  
Chicago, IL 60610

# About Example 6.5

- This example uses `xsl:if` to assure that missing optional elements do not trigger unwanted output.
- In this case, the `test` attribute is checking whether an element exists that fits this XPATH expression.
- `xsl:for-each` is used to assure that we generate the proper number of lines for street address in the output tree.
- Using a `period` for the value of the `select` attribute selects the text of the current element.
- Note the use of relative XPATH expressions to retrieve the parts of the address.

# 6.6.xslt: Finishing Touches

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xhtml"
    doctype-system=
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
    encoding="UTF-8" indent="yes"/>

  <xsl:template match="/">
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
        <title>Employee List</title>
      </head>
      <body>
        <h2>Xampul Employee List</h2>
        <h3><xsl:value-of select="/employeeelist/listregion"/>
          Region</h3>
        <xsl:for-each select="employeeelist/employee">
          <p>
            <xsl:value-of select="name/firstname"/>
```

## 6.6.xslt: Finishing Touches (cont.)

```
<xsl:text> </xsl:text>
<xsl:if test="name/middleinit">
  <xsl:value-of select="name/middleinit"/>
  <xsl:text>. </xsl:text>
</xsl:if>
<xsl:value-of select="name/lastname"/>
<xsl:if test="name/suffix">
  <xsl:text> </xsl:text>
  <xsl:value-of select="name/suffix"/>
</xsl:if>
<br/>
<xsl:for-each select="address/street">
  <xsl:value-of select="."/>
  <br/>
</xsl:for-each>
<xsl:value-of select="address/city"/>
<xsl:text>, </xsl:text>
<xsl:value-of select="address/state"/>
<xsl:text> </xsl:text>
<xsl:value-of select="address/zip"/>
```

## 6.6.xslt: Finishing Touches (last)

```
<xsl:for-each select="phone">
  <br/>
  <xsl:value-of select="."/>
  <xsl:text> (</xsl:text>
  <xsl:value-of select="@type"/>
  <xsl:text>)</xsl:text>
</xsl:for-each>
</p>
</xsl:for-each>
<br/>
<br/>
<br/>
<i>last revised on <xsl:value-of
  select="/employee/listdate"/>
</i>
</body>
</html>
</xsl:template>

</xsl:stylesheet>
```



# Xampul Employee List

## Central Region

Xavier Ampul III  
1234 Main Street  
Apartment 401  
Chicago, IL 60610  
555-555-5555 (home)

Frances R. Smith  
559 Primary Avenue  
Evanston, IL 60201  
555-555-5590 (home)  
555-555-5591 (mobile)

Raymond Jones  
987 Center Street  
Chicago, IL 60610  
555-555-0987 (mobile)

*last revised on April 4, 2006*

# About Example 6.6

- `xs1:for-each` is used in this example to control the number of `phone` lines generated in the output tree.
- The XPATH expression `@type` is used to retrieve the value of the attribute `type` for the current `phone` element.
- The retrieval and output of the `listdate` element has been moved from the top of the page to the bottom of the page to demonstrate the ease of processing elements in an arbitrary order with XSLT.

# 6.6a.xml: Add Namespace

```
<?xml version="1.0" encoding="utf-8"?>
<employeelist
  xmlns="http://www.ligent.net/namespaces/examples/employeelist">

  <!-- This is a copy of ex_6.6.xml with a
    namespace added -->

  <listdate>April 4, 2006</listdate>
  <listregion>Central</listregion>

  <employee>
    <name>
      <firstname>Xavier</firstname>
      <lastname>Ampul</lastname>
      <suffix>III</suffix>
    </name>
    <address>
      <street>1234 Main Street</street>
      <street>Apartment 401</street>
      <city>Chicago</city>
```

## 6.6a.xml: Add Namespace (cont.)

```
<state>IL</state>  
<zip>60610</zip>  
</address>  
<phone type="home">555-555-5555</phone>  
</employee>
```

```
<employee>  
  <name>  
    <firstname>Frances</firstname>  
    <middleinit>R</middleinit>  
    <lastname>Smith</lastname>  
  </name>  
  <address>  
    <street>559 Primary Avenue</street>  
    <city>Evanston</city>  
    <state>IL</state>  
    <zip>60201</zip>  
  </address>  
  <phone type="home">555-555-5590</phone>  
  <phone type="mobile">555-555-5591</phone>
```

## 6.6a.xml: Add Namespace (last)

```
</employee>
```

```
<employee>
```

```
  <name>
```

```
    <firstname>Raymond</firstname>
```

```
    <lastname>Jones</lastname>
```

```
  </name>
```

```
  <address>
```

```
    <street>987 Center Street</street>
```

```
    <city>Chicago</city>
```

```
    <state>IL</state>
```

```
    <zip>60610</zip>
```

```
  </address>
```

```
  <phone type="mobile">555-555-0987</phone>
```

```
</employee>
```

```
</employeelist>
```

# Xampul Employee List

**Region**

*last revised on*

# About Example 6.6a Without Namespace

- XSLT processors are **namespace-aware**.
- XSLT stylesheets must reference the namespace of the input content document in order to access content.

# 6.6a.xslt: Adjust For Namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xpath-default-namespace=
  "http://www.ligent.net/namespaces/examples/employeelist"
  >

<xsl:output method="html"
  doctype-system=
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
  encoding="UTF-8" indent="yes"/>

<xsl:template match="/">
  <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
      <title>Employee List</title>
    </head>
    <body>
      <h2>Xampul Employee List</h2>
      <h3><xsl:value-of select="/employeelist/listregion"/>

```



## 6.6a.xslt: Adjust For Namespace (cont.)

```
Region</h3>
<xsl:for-each select="employeelist/employee">
  <p>
    <xsl:value-of select="name/firstname"/>
    <xsl:text> </xsl:text>
    <xsl:if test="name/middleinit">
      <xsl:value-of select="name/middleinit"/>
      <xsl:text>. </xsl:text>
    </xsl:if>
    <xsl:value-of select="name/lastname"/>
    <xsl:if test="name/suffix">
      <xsl:text> </xsl:text>
      <xsl:value-of select="name/suffix"/>
    </xsl:if>
    <br/>
    <xsl:for-each select="address/street">
      <xsl:value-of select="."/>
      <br/>
    </xsl:for-each>
    <xsl:value-of select="address/city"/>
```

## 6.6a.xslt: Adjust For Namespace (cont.)

```
<xsl:text>, </xsl:text>
<xsl:value-of select="address/state"/>
<xsl:text> </xsl:text>
<xsl:value-of select="address/zip"/>
<xsl:for-each select="phone">
  <br/>
  <xsl:value-of select="."/>
  <xsl:text> (</xsl:text>
  <xsl:value-of select="@type"/>
  <xsl:text>)</xsl:text>
</xsl:for-each>
</p>
</xsl:for-each>
<br/>
<br/>
<br/>
<i>last revised on <xsl:value-of
  select="/employee/listdate"/>
</i>
</body>
```

## 6.6a.xslt: Adjust For Namespace (last)

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

# Xampul Employee List

## Central Region

Xavier Ampul III  
1234 Main Street  
Apartment 401  
Chicago, IL 60610  
555-555-5555 (home)

Frances R. Smith  
559 Primary Avenue  
Evanston, IL 60201  
555-555-5590 (home)  
555-555-5591 (mobile)

Raymond Jones  
987 Center Street  
Chicago, IL 60610  
555-555-0987 (mobile)

*last revised on April 4, 2006*

# About Example 6.6a With Namespace

- Add the `xpath-default-namespace` attribute to the `xs1:stylesheet` element to make input document content accessible to the XSLT stylesheet.