XSLT Vortrag

1. Was ist XSLT?

XSLT ist eine Transformationssprache für XML Dokumente.
XSLT war wahrscheinlich die erste XML basierte Programmiersprache.
XSLT ist von der Struktur her Template-Matching, nicht Anweisungsorientiert
XSLT hat nur minimalen Zugriff auf das Betriebssystem, und kann daher wenig Schaden anrichten.

WORKFLOW DIAGRAMM

2. Beispiel XML

```
<ADDRESS>
 <VORNAME>Philipp</VORNAME>
 <NAME>Gühring</NAME>
 <STREET>Hebenstreitstr. 16</STREET>
 <ZIP>2602</ZIP>
 <CITY>Neurißhof</CITY>
 <COUNTRY>Austria</COUNTRY>
 <TEL>+43-2628-49103</TEL>
 <FAX/>
 <MOBILE>+43-664-9953109</MOBILE>
 <EMAIL>pg@futureware.at</EMAIL>
 <URL>http://www.futureware.at/</URL>
</ADDRESS>
```

3. Anwendungsmöglichkeiten von XSLT

Visualisierung der Daten in Tabellenform

Vorname	Name	Straße	PLZ	Ort	Land	Telefon
Philipp	Gühring	Hebenstreit	2602	Neurißhof	Austria	+43-2628-...

Visualisierung derselben Daten in Etikettenform:

Philipp Gühring
Hebenstreitstr. 16
A-2602 Neurißhof

4. XHTML

- <?xml version="1.0" encoding="iso-8859-1"?>
- Alle Tags schließen, Tag Struktur bereinigen
- Kleinschreibung der Tags
- Anführungszeichen der Tag Attribute
- & => & und < => <

tidy -asxml altes.html >neues.xhtml

SVG, MathML, RDF, Docbook, BMEcat, openTRANS, OpenOffice, KOffice, ...

5. Homepage

```
<?xml version="1.0" encoding="iso-8859-1"?>
<homepage>

 <!-- Hier fangen wir mit dem Menü der Homepage und den Inhalten an -->
 <content>
  <menu>
   <name>LivingXML</name>
   <text>
    <h1>LivingXML</h1>
    Willkommen auf der LivingXML Homepage
   </text>
  </menu>
  <menu id="dv">
   <name>Datenverarbeitung</name>
   <text>XML ist ein leistungsfähiges Datenverarbeitungs System</text>
  </menu>
 </content>

 <!-- Und hier sind die Neuigkeiten -->
 <news>
  <artikel>
   <headline>XML Vortrag</headline>
   <link>http://livingxml.net/</link>
  </artikel>
  <artikel>
   <headline>XSLT Vortrag bei der Jugat</headline>
   <link>http://www.jugat.at/</link>
  </artikel>
  <artikel>
   <headline>LivingXML bei den Linuxwochen 2003</headline>
   <link>http://www.linuxwochen.at/</link>
  </artikel>
 </news>
</homepage>
```

6. XSLT

XSLT ist die erste XML Programmiersprache.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" encoding="utf-8"/>
 <xsl:param name="menu"/>
 <xsl:param name="news"/>

<xsl:template match="/">
<html>
 <head>
  <title>&lt;LivingXML.net&gt;</title>
```

```
 </head>

<body>
<table>


<!-- Menü -->
<tr>
<xsl:for-each select="/homepage/content/*">
<td><a href="?menu={position()}"><xsl:value-of select="name"/></a></td>
</xsl:for-each>
</tr>
<tr><td>

<!-- News Ticker -->
<h4>News Ticker</h4>

<xsl:for-each select="/homepage/news/*">
 <a>
 <xsl:choose>
  <xsl:when test="link">
   <xsl:attribute name="href"><xsl:value-of select="link"/></xsl:attribute>
  </xsl:when>
  <xsl:otherwise>
   <xsl:attribute name="href">?news=<xsl:value-of select="position()"/></xsl:attribute>
  </xsl:otherwise>
 </xsl:choose>
 <b><xsl:value-of select="headline"/></b>
 </a>
 <br/>
</xsl:for-each>

</td>
<td colspan="10">

<!-- Seiten Inhalt -->
<br/>
<xsl:choose>
 <xsl:when test="$news">
  <xsl:copy-of select="/homepage/news/artikel[number($news)]/text"/>
 </xsl:when>
 <xsl:when test="$menu">
  <xsl:copy-of select="/homepage/content/menu[number($menu)]/text"/><br/><br/>
  <xsl:if test="/homepage/content/menu[number($menu)+1]/name">
   <a href="?menu={$menu+1}">&gt;&gt;
   <xsl:value-of select="/homepage/content/menu[number($menu)+1]/name"/></a>
  </xsl:if>
 </xsl:when>
 <xsl:otherwise>
  <xsl:copy-of select="/homepage/content/menu[1]/text"/>
  <a href="?menu=2">&gt;&gt; <xsl:value-of
select="/homepage/content/menu[2]/name"/></a>·
 </xsl:otherwise>
</xsl:choose>

</td>
```

```
</tr>
</table>

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

7. Templates

```
<?xml version='1.0' encoding='iso-8859-1'?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" encoding="utf-8"/>
 <xsl:param name="menu"/>
 <xsl:param name="news"/>

<xsl:template match="/">
<html>
 <head>
  <title>&lt;LivingXML.net&gt;</title>
 </head>

<body>
<table>


<!-- Menü -->
<tr>
<xsl:for-each select="/homepage/content/*">
<td><a href="?menu={position()}"><xsl:value-of select="name"/></a></td>
</xsl:for-each>
</tr>
<tr><td>

<!-- News Ticker -->
<h4>News Ticker</h4>

<xsl:for-each select="/homepage/news/*">
 <a>
 <xsl:choose>
  <xsl:when test="link">
   <xsl:attribute name="href"><xsl:value-of select="link"/></xsl:attribute>
  </xsl:when>
  <xsl:otherwise>
   <xsl:attribute name="href">?news=<xsl:value-of select="position()"/></xsl:attribute>
  </xsl:otherwise>
 </xsl:choose>
 <b><xsl:value-of select="headline"/></b>
 </a>
 <br/>
</xsl:for-each>

</td>
```

```
<td colspan="10">

<!-- Seiten Inhalt -->
<br/>
<xsl:choose>
 <xsl:when test="$news">
  <xsl:copy-of select="/homepage/news/artikel[number($news)]/text"/>
 </xsl:when>
 <xsl:when test="$menu">
  <xsl:copy-of select="/homepage/content/menu[number($menu)]/text"/><br/><br/>
  <xsl:if test="/homepage/content/menu[number($menu)+1]/name">
   <a href="?menu={$menu+1}">&gt;&gt;
   <xsl:value-of select="/homepage/content/menu[number($menu)+1]/name"/></a>
  </xsl:if>
 </xsl:when>
 <xsl:otherwise>
  <xsl:copy-of select="/homepage/content/menu[1]/text"/>
  <a href="?menu=2">&gt;&gt; <xsl:value-of
select="/homepage/content/menu[2]/name"/></a>·
 </xsl:otherwise>
</xsl:choose>

</td>
</tr>
</table>

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

8. XPath

XPath wird die Sprache genannt, mit der man in XML Daten navigieren kann, und Abfragen machen kann:

```
<xsl:value-of select=" /Dokumente/Rechnungen/Rechnung[Header/Version='1.0' and
Footer/Count>5]/Buyer/Address/* "/>
```

Der ganze Ausdruck, der als Argument select angegeben wird, bedeutet, daß von allen Dokumenten alle Rechnungen herausgesucht werden sollen, bei denen die Version im Header 1.0 ist, und die mehr als 5 Rechnungszeilen haben, und gibt alle Adressdaten des Käufers aus.

9. Die wichtigsten XSLT "Befehle"

9.1. xsl:value-of

Mit xsl:value-of kann man einen einzelnen Wert aus einem Feld, oder Attribut ausgeben:

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

gibt Philipp aus.

9.2. xsl:copy-of

xsl:copy-of kopiert einen gesamten XML Sub-Baum, im Gegensatz zu xsl:value-of, daß nur die Texte extrahiert, aber keine Subelemente mitnimmt:

```
<aktuell>
 ...
</aktuell>
```

```
<history>
  <xsl:copy-of select="/aktuell"/>
  <xsl:copy-of select="/history/*"/>
</history>
```
Mit einem Konstrukt dieser Art kann man beim verändern eines Datensatzes auf einfache Weise den alten Datensatz im History Block archivieren.

9.3. xsl:for-each

Mit xsl:for-each macht man eine Schleife über alle Elemente, die die XPath Abfrage liefert:
```
<xsl:for-each select="/ADDRESS/*">
  <xsl:value-of select="name()"/>: <xsl:value-of select="."/>
</xsl:for-each>
```
Das ist eine Schleife über alle Felder im ADDRESS Block, und in der Schleife wird jeweils der Name des Feldes und der Inhalt ausgegeben:
VORNAME: **Philipp**
NAME: **Gühring**
STREET: **Hebenstreitstr. 16**
ZIP: **2602**
CITY: **Neurißhof**
COUNTRY: **Austria**
...

9.4. xsl:if

```
<xsl:if test="VORNAME='Philipp'">
Lange nicht gesehen, Philipp. Wie wäre es mit einer Partie Schach?
</xsl:if>
```

9.5. xsl:choose

Mit xsl:choose kann man Abfragen machen, es wird auch für if-else Konstrukte verwendet, nachdem is kein else gibt bei XSLT.
```
<xsl:choose>
  <xsl:when test=".='Gold'">
    Wir haben Gold gefunden! Jubel!
  </xsl:when>
  <xsl:when test=".='Silber'">
    Wir haben Silber gefunden!
  </xsl:when>
  <xsl:otherwise>
    Ich habe keine Ahnung, was das sein soll ...
  </xsl:otherwise>
</xsl:choose>
```

9.6. xsl:variable

Mit xsl:variable kann man Variablen definieren, die ab dann in derselben und allen tieferen Ebenen bekannt sind:

```
<xsl:variable name="myvar">
Hallo Welt!
</xsl:variable>
```

Variablen spricht man mit $ davor an:

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

Gibt dann Hallo Welt! aus.

9.7. xsl:element

Mit xsl:element kann man sich eigene XML Elemente selber zusammenbauen:
<xsl:element name="table">
<xsl:attribute name="width">100%</xsl:attribute>
<tr><td/></tr>
</xsl:element>
Dadurch kann man Elemente und deren Attribute bei Bedarf dynamisch generieren.

10. XML Anwendungen

Steuerung: Modell-View-Controller Konzept

Modell: XML
View: XSLT
Controller: Perl/PHP/Java/Workflow + XSLT

Skalierbarkeit: XSLT > 1 MB
Dateisystem
Jeder Datensatz (optimale Größe: 1 KB) kommt als XML Dokument in eine Datei.
Alle Datensätze einer "Tabelle" kommen in ein Verzeichnis.
Eine Million Dateien in einem Verzeichnis?

adr/1239125:
<ADDRESS>
<VORNAME>**Philipp**</VORNAME>
<NAME>**Gühring**</NAME>
...

Indizierung: Symbolische Links
ind_adr/Philipp_Gühring_1239125 -> ../sta/1239125

Mehrere Felder? Mehrere gleiche Indizes?

11. Referenzen

http://www.xml.com/
W3C: http://www.w3.org/
Selfhtml: http://selfhtml.teamone.de/
Sablotron (XSLT Parser): http://www.gingerall.cz/
Java XSLT Parser: Saxon, Xerces
LivingXML: http://www.livingxml.net/