

# Clientseitige Verarbeitung von XML (WLMML)-Dateien mit XSLT und JavaScript

S. Paar, O. Strehl, H. D. Quednau

Fachgebiet für Biometrie und Angewandte Informatik Department für Ökosystem- und Landschaftsmanagement am Wissenschaftszentrum Weihenstephan für Ernährung, Landnutzung und Umwelt der Technischen Universität München

## Zusammenfassung

Die an LMML (Learning Material Markup Language) der Universität Passau (IFIS - Institut für Informationssysteme und Softwaretechnik) angelehnte XML-Sprache WLMML (WELPE-LMML, Weihenstephaner E-Learning-Projekte) ist als vereinfachte Version von LMML konzipiert und ermöglicht es Lerninhalte strukturiert abzubilden.

Zur clientseitigen Verarbeitung von XML (WLMML)-Dateien werden im WELPE-Framework XSLT (Extensible Stylesheet Language Transformations) und JavaScript herangezogen.

XSLT erlaubt es z. B. über die Auswahl bestimmter Elemente aus einem XML-Quelldokument ein verändertes Zieldokument zu erzeugen. Das Ergebnis kann z. B. ein weiteres XML- oder ein HTML-Dokument sein.

JavaScript ist eine meist clientseitige, objektbasierte Skriptsprache, die vom Interpreter des Browsers abgearbeitet wird. Die Grundlagen von JavaScript sind unter dem Namen ECMAScript zum ISO-Standard erhoben worden. JavaScript bietet die Möglichkeit der komplexeren Verarbeitung von XML mit XSLT.

Die im WELPE-Framework abgelegten Lerninhalte sind in Standard-Browsern (z. B. Internet Explorer 6 bzw. 7 oder Mozilla Firefox 1.5 bzw. 2) ohne Installation und im Grundsatz auch ohne Verbindung zu einem Internet/Intranet-Server lauffähig (inklusive z. B. der Unterstützung der Mehrsprachigkeit oder einer Volltext-Suchmaschine). Der Grund dafür ist, dass nach einem Download der Lerninhalte bei der Verarbeitung der WLMML-Dateien mit Hilfe von XSLT und JavaScript (WELPE-Framework) keine Datenübertragung zwischen einem Server und Client (Browser) stattfinden muss (clientseitige Verarbeitung).

## Abstract

The XML language WLMML (WELPE-LMML, Weihenstephaner E-Learning-Projekte) modelled on LMML (Learning Material Markup Language, developed by IFIS - Institute for Information Systems and Software Engineering, University of Passau) is a simplified version of LMML and allows to show contents in a structured way.

In order to process XML (WLMML) files in the WELPE framework XSLT (Extensible Stylesheet Language Transformations) and JavaScript is used.

XSLT allows to create an altered document by means of choosing certain elements from an original XML document. The result may be another XML or HTML document.

JavaScript is a client-based object-oriented scripting language which is processed by the interpreter of the browser. The basics of JavaScript were made an ISO-standard under the name ECMA. JavaScript enables a more complex processing of XML with XSLT.

The WELPE framework e-learning contents are supported by common browsers (e. g. Internet Explorer 6/7 or Mozilla Firefox 1.5/2) without an installation and principally without a connection to an internet/intranet-server (including e. g. the service of multi languages and searching of words). The reason is, that after loading the e-learning contents a data-transfer between server and client (browser) is not necessary when processing XML (WLMML) files with the aid of XSLT and JavaScript (client-based processing).

# WLMML (Weihenstephaner Learning Material Markup Language)

## Einführung

Mit der bereits etablierten Extensible Markup Language (XML) (W3C 2004) steht auch im Bereich E-Learning ein reines Text-Datenformat für die Erfassung, den Austausch und die Speicherung von Lerninhalten zur Verfügung. Als Metasprache bietet XML die Möglichkeit eigene Sprachen über ein Schema (W3C 2002) zu definieren. Die an LMML (Learning Material Markup Language) (LMML 2006, PAAR 2004) der Universität Passau angelehnte Sprache WLMML (WELPE-LMML, Weihenstephaner E-Learning-Projekte) stellt ein Beispiel dafür dar. Sie ist als vereinfachte Version von LMML konzipiert und erlaubt es Lerninhalte in strukturierter Form abzubilden. Modularisierung und Wiederverwendung von Lehrmaterial stellen wie auch bei LMML weitere Ziele dar.

WLMML wurde entwickelt, um

- den Einsatz neuer Elemente und Attribute (auch alter LMML-Elemente und -Attribute) an spezifischen Stellen im Dokument zu ermöglichen,
- die 8 LMML-Schema-Dateien durch eine vereinfachte einzige Schema-Datei zu ersetzen,
- den Einsatz von z. B. Microsoft Word 2003 (bzw. 2007) oder Altova XMLSpy 2006 (bzw. 2007) bei der Erstellung von WLMML-Dateien (mit dem identischen WLMML-Schema) zu ermöglichen.

In WLMML werden nicht alle Elemente und Attribute von LMML übernommen und nur einige Elemente bzw. Attribute neue eingeführt.

## Schema

Eine Schema-Datei gibt eine formale Grammatik an, in der die Bezeichnungen der in Dokumenteninstanzen zulässigen Elemente mit ihren Attributen und deren Verschachtelung definiert sind. Sie ist selbst eine XML-Datei und kann von beliebigen Texteditoren erstellt und bearbeitet werden.

Das Modell von WLMML hält sich in engem Rahmen an das Modell von LMML (SÜSS 2000, LMML 2006). Der Aufbau (Quelltext) der einzigen WLMML-Schema-Datei, die ein Abbild des WLMML-Modells darstellt, unterscheidet sich allerdings in hohem Masse vom Aufbau der 8 LMML-Schema-Dateien (LMML 2006).

Grundsätzlich wird Lehrmaterial in WLMML (wie im LMML) in Inhaltsabschnitte (Element section) strukturiert und durch Inhaltselemente (Elemente definition, detail, example ...) ergänzt. Innerhalb der Inhaltselemente können gliedernde Strukturobjekte (Elemente ul, ol, table) und allgemeine Medienobjekte (Elemente image, sound, text ...) über das Inhaltselement paragraph eingebunden werden.

Abbildung 1 zeigt einen Ausschnitt aus der WLMML-Schema-Datei mit verschiedenen Elementen, die in einem WLMML-Dokument eingesetzt werden können.

Dabei wird z. B. deutlich, dass im Root-Element wlmml nur zwei Elemente bibliography (Literaturverzeichnis) und section (Inhaltsabschnitt) erlaubt sind. Das Element section wiederum kann die Elemente definition (Definition), detail (Vertiefung), example (Beispiel), exercise (Übung), learningObjective (Lernziel), paragraph (Absatz) und summary (Zusammenfassung) beinhalten.

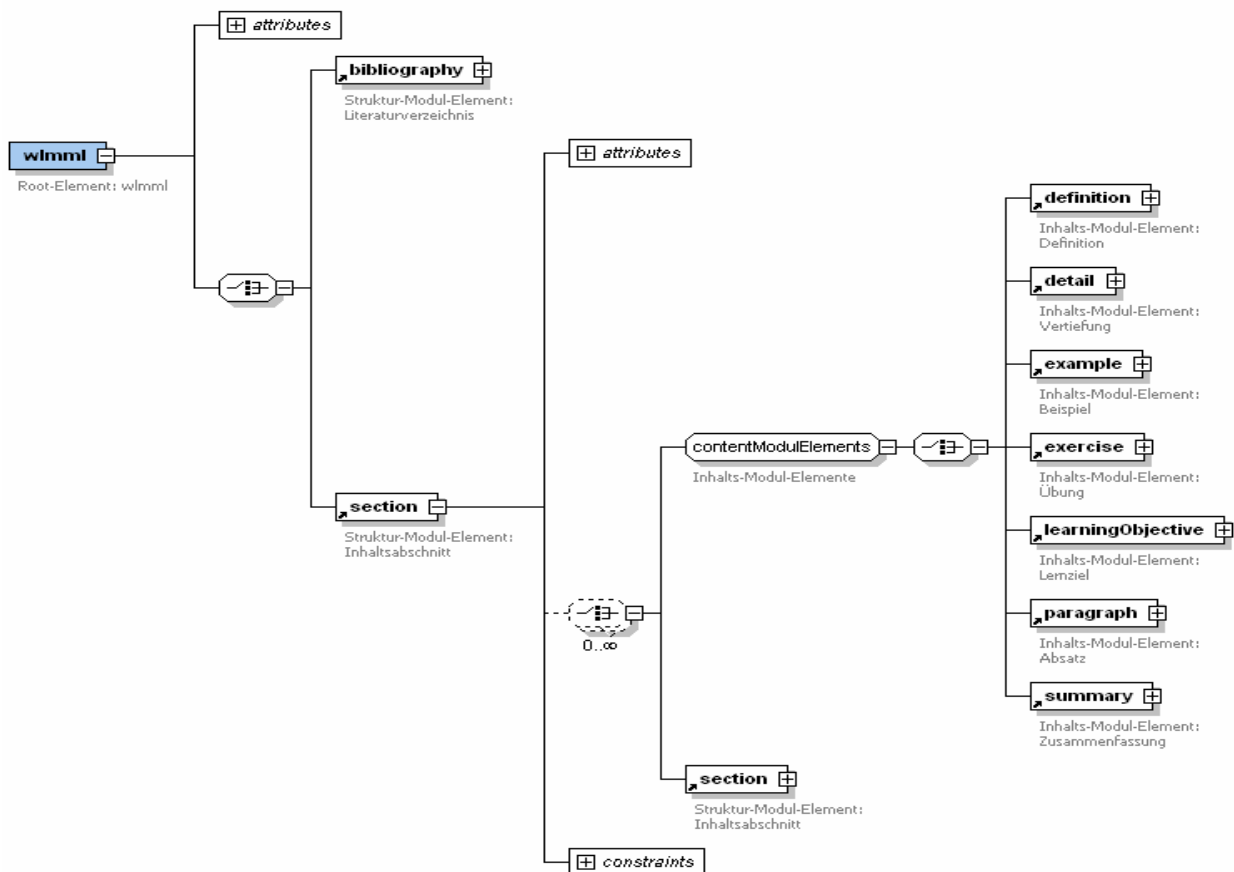


Abbildung 1: Ausschnitt aus dem WLMML-Schema

## WLMML in einem XML Beispiel

Neben herkömmlichen XML-Editoren (bzw. Text-Editoren) eignen sich z. B. Microsoft Word 2003 und 2007, Altova XMLSpy oder Eclipse (Web Tools Platform) gut WLMML-Dateien zu erstellen.

Folgende Abbildung 2 zeigt den Quelltext einer WLMML-Datei. Im Prolog der XML-Datei ist ein in XML üblicher internationaler Unicode-Zeichensatz (UTF-8) eingetragen. Damit dürfen z. B. die deutschen Zeichen ä, ö, ü im Fließtext der Lerneinheit direkt angegeben werden. Die Datei muss allerdings im Unicode-Format und nicht im ANSI-Format (American National Standards Institute, unter Windows-Betriebssystemen üblich) abgespeichert werden. In der zweiten Zeile wird auf eine XSLT-Datei zur Transformation in eine Hypertext Markup Language (HTML) (W3C 1999 b) verwiesen. Anschließend taucht das root-Element wlmml mit u. a. dem Verweis auf die WLMML-Schemadatei auf. Das folgende Element section umschließt einen Absatz (paragraph) mit Textinhalt (text) und einer Grafik (image).

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="../../system/xsl/main.xsl"?>
<wlmml languageSystem="de" xmlns="wlmml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="wlmml ..\..\system\xsd\wlmml\wlmml.xsd">
    <section title="Beispiel">
        <paragraph label="1">
            <text>Beispiel</text>
            <image title="Himalaya" uri="../../media/10003_00003_himalaya.jpg"/>
        </paragraph>
    </section>
</wlmml>

```

Abbildung 2: Quelltext einer WLMML-Datei

## XSLT

Zur Verarbeitung von XML-Dateien wird häufig die Extensible Stylesheet Language Transformations (XSLT) (W3C 1999 a) herangezogen. Sie erlaubt es z. B. über die Auswahl bestimmter Elemente aus einem XML-Quelldokument ein verändertes Zieldokument zu erzeugen. Das Transformationsergebnis kann ein weiteres XML- oder ein HTML-Dokument sein.

XSLT kann sowohl client- als auch server-seitig eingesetzt werden. In den weltweit verbreiteten Browsern Microsoft Internet Explorer 6 und 7, sowie Mozilla Firefox 1.5 und 2 ist die Verarbeitung von XSLT implementiert.

Durch die Zuweisung von HTML-Elementen (z. B. Überschrift-Element h1) über die XSLT-Transformation werden Textteile formatiert. Mit Hilfe von Cascading Style Sheets (CSS) (W3C 1996) können die Formatierungsanweisungen verfeinert und zentral verwaltet werden. Auf die Formatierung von XSL-FO (W3C 2001) wird verzichtet, da diese Sprache von keinem Browser mit einem nennenswerten Marktanteil umgesetzt wird (siehe z. B. GAGARINOV & IVERSON 2005).

Von besonderer Bedeutung für die Verarbeitung mehrerer XML-Dateien (wie im WELPE-Framework) ist die XSLT-Funktion document() (z. B. in Abbildung 9). Sie ist in der Lage sukzessive verschiedene XML-Dateien zu laden und den Inhalt anderen XSLT-Funktionen zur Verfügung zu stellen.

Abbildung 3 zeigt eine im Browser geöffnete WLMML-Datei (Quelltext in Abbildung 2). Die zur Transformation und erweiterten Funktionsunterstützung eingesetzten Dateien (u. a. XSLT-, CSS- und JavaScript-Dateien) müssen vorhanden sein (siehe Kapitel Clientseitige Verarbeitung im WELPE-Framework).



Abbildung 3: Anzeige einer WMML-Datei (Endergebnis der XSLT-Transformation) im MS Internet Explorer 7

## JavaScript

JavaScript ist eine objektbasierte Skriptsprache, die meist clientseitig vom Interpreter des Browsers abgearbeitet wird. Die grundlegenden Teile von JavaScript sind unter der Bezeichnung ECMAScript (The European Computer Manufacturers Association) zum ISO-Standard erhoben worden (ECMA 1999). JavaScript ist in fast allen gängigen Browsern implementiert und bietet die Möglichkeit der komplexeren Verarbeitung von XML mit XSLT. Aus Sicherheitsgründen besteht keine Möglichkeit (abgesehen von Cookies) auf dem Client-Rechner Dateien zu erstellen oder zu speichern.

JavaScript erlaubt es z. B. XML- und XSLT-Dateien zu laden und zusammen mit Parameterwerten an den XSLT-Prozessor des Browsers zu übergeben. Das Transformationsergebnis wird wieder entgegengenommen und kann am Bildschirm dargestellt werden. Als Schnittstelle zum XSLT-Prozessor des Browsers ergeben sich mit JavaScript vielfältige Möglichkeiten der Verarbeitung von XML-Dateien.

# Clientseitige Verarbeitung im WELPE-Framework

## Einführung

Im WELPE-Framework werden folgende internationale Standards/Normen und Spezifikationen eingesetzt:

- XML
- Schema
- HTML
- XSLT
- JavaScript
- CSS
- SCORM (Sharable Content Objekt Reference Model) (SCORM 2006)

THROPP (2004) beschreibt weitere SCORM (2004) unterliegende Standards und Spezifikationen (u. a. Instructional Management System Content Packaging (IMS CP 2003), Learning Object Metadata (LOM) (IEEE LTSC 2005)). Durch die konsequente Umsetzung dieser Spezifikationen (bzw. Standards/Normen) im WELPE-Framework soll die zukunftssichere Verwendung (bzw. Austausch) von Lerninhalten sichergestellt werden (siehe auch STREHL & QUEDNAU 2005).

Grundsätzlich ist das WELPE-Framework keine Lernplattform (WIKIPEDIA 2006) an sich, sondern bietet mit seiner inhaltlichen Prägung (WLMML-Dateien) eine Möglichkeit Lerninhalte abzubilden. Es besteht aus verschiedenen Ordnern und Dateien (XML-, XSLT-, HTML-, CSS-, JavaScript-Dateien) (Abbildung 4), die eine clientseitige Verarbeitung erlauben. Auch wenn die WELPE-Framework-Dateien z. B. ursprünglich auf einem Server liegen, werden sie über den Aufruf der WLMML-Dateien (Lerninhalte) im Browserfenster auf den Client-Rechner geladen und dort für die Verarbeitung der WLMML-Dateien eingesetzt.

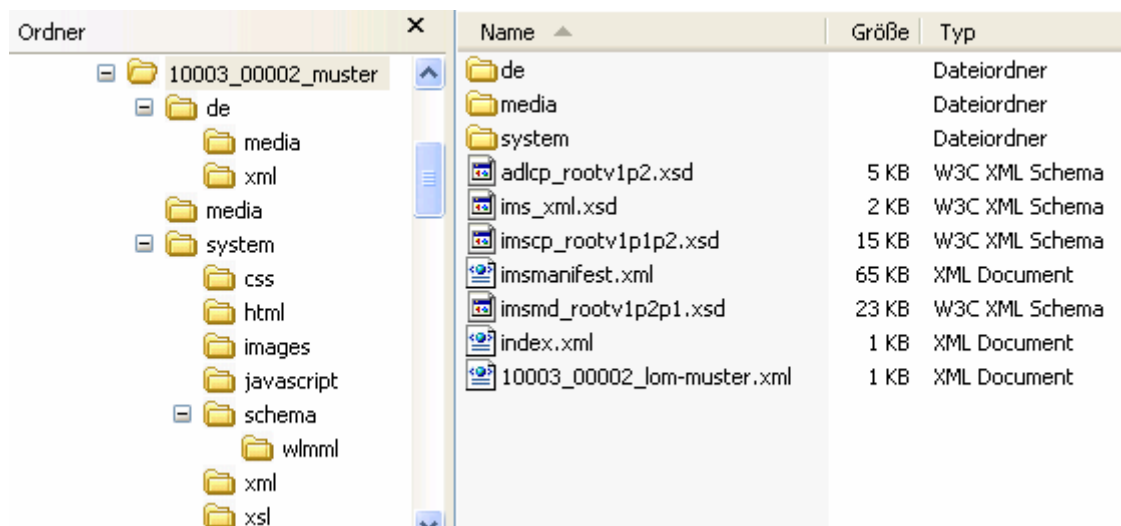


Abbildung 4: Verzeichnisstruktur des WELPE-Frameworks

Die fertigen XML-(WLMML)-Dateien werden im WELPE-Framework SCORM-konform in einer `imsmanifest.xml` aufgelistet, mit Hilfe von XSLT und JavaScript clientseitig abgearbeitet und im Browser zur Anzeige gebracht. Die `imsmanifest.xml` stellt innerhalb von SCORM eine Art Inhaltsverzeichnis der Lerneinheit dar (mit Pfadangabe zu den einzelnen Dateien = Ressourcen). Als SCORM-Grundbaustein wird die `imsmanifest.xml` im WELPE-Framework wie auch in SCORM-kompatiblen Lernplattformen vorausgesetzt. Im WELPE-Framework spiegelt sich der Inhalt der `imsmanifest.xml` u. a. als Navigationsstruktur im linken Teil des Browserfensters wieder (Abbildung 6) und wird als Auflistung der in der Volltextsuche zu durchsuchenden XML-Dateien benötigt (Abbildung 9).

Die im WELPE-Framework abgelegten Lerninhalte sind in Standard-Browsern wie z. B. Internet Explorer 6 bzw. 7 oder Mozilla Firefox 1.5 bzw. 2 lauffähig. Es ist keine Installation und im Grundsatz auch keine Verbindung zu einem Internet/Intranet-Server nötig (bei voller Funktionstauglichkeit z. B. der Mehrsprachigkeit oder der Volltext-Suchmaschine). Der Grund dafür ist (neben dem Einsatz internationaler Spezifikationen/Standards), dass nach einem Download der Lerninhalte bei der Verarbeitung der WLMML-Dateien keine Datenübertragung zwischen einem Server und Client (Browser) stattfinden muss (clientseitige Verarbeitung). Die identischen Lehrinhalte können damit sowohl online über eine Lernplattform, über einen Web-Server oder offline (nach dem Download) im Browser betrachtet werden.

### **Beispiel clientseitiger Verarbeitung (Volltext-Suchmaschine)**

Am Beispiel der im WELPE-Framework eingebundenen Volltext-Suchmaschine wird im Folgenden der Ablauf einer Volltext-Suche mit den wichtigsten involvierten XSLT- und JavaScript-Dateien geschildert.

Abbildung 5 beleuchtet in einer Übersicht grafisch den Ablauf der Verarbeitung anhand der wichtigsten beteiligten Dateien. Ausgehend von der `index_sel.xml` wird über die Betätigung der Suche-Schaltfläche die JavaScript-Funktion `xslt()` in der `xslt.js` aufgerufen. Als Parameter werden unter anderem die zu öffnenden XSLT-Dateien und der Suchbegriff übergeben. Die erste XSLT-Datei (`index_search_xml.xml`) nimmt den Suchbegriff entgegen und führt die Suche in den in der `imsmanifest.xml` aufgeführten XML-Dateien durch. Die Ergebnisliste wird von der `index_search_html.xml` aufgenommen und in HTML transformiert. Abschließend wird der von der `index_search_html.xml` erzeugte HTML-Quellcode über Funktionen in der `xslt.js` im Browser zur Anzeige gebracht.

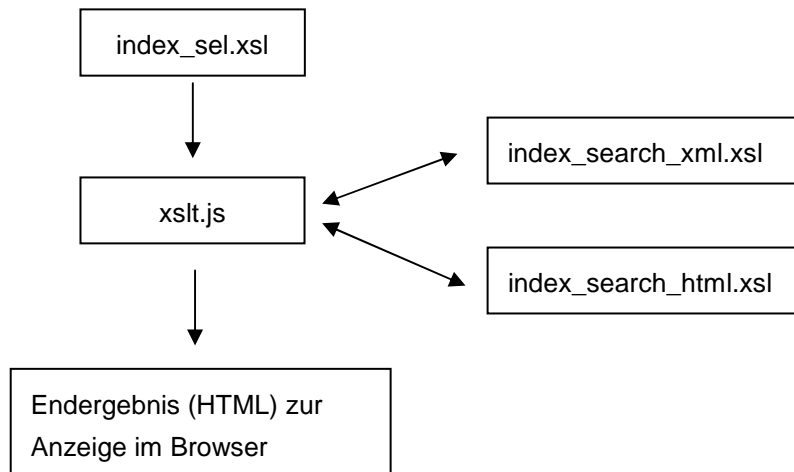


Abbildung 5: Grafische Übersicht des Ablaufes einer Volltext-Suche anhand der wichtigsten beteiligten Dateien

Geht man genauer auf die in Abbildung 5 aufgeführten Dateien ein, ergibt sich folgendes Bild. Ruft der Benutzer die Inhaltssuche auf, sorgt die index\_sel.xml im oberen Bereich des Browserfensters u. a. für die Anzeige des Such-Formularfeldes. Wurde ein Suchbegriff eingegeben, kann über die Enter-Taste oder die Schaltfläche Suche die Suche gestartet werden (Abbildung 6).

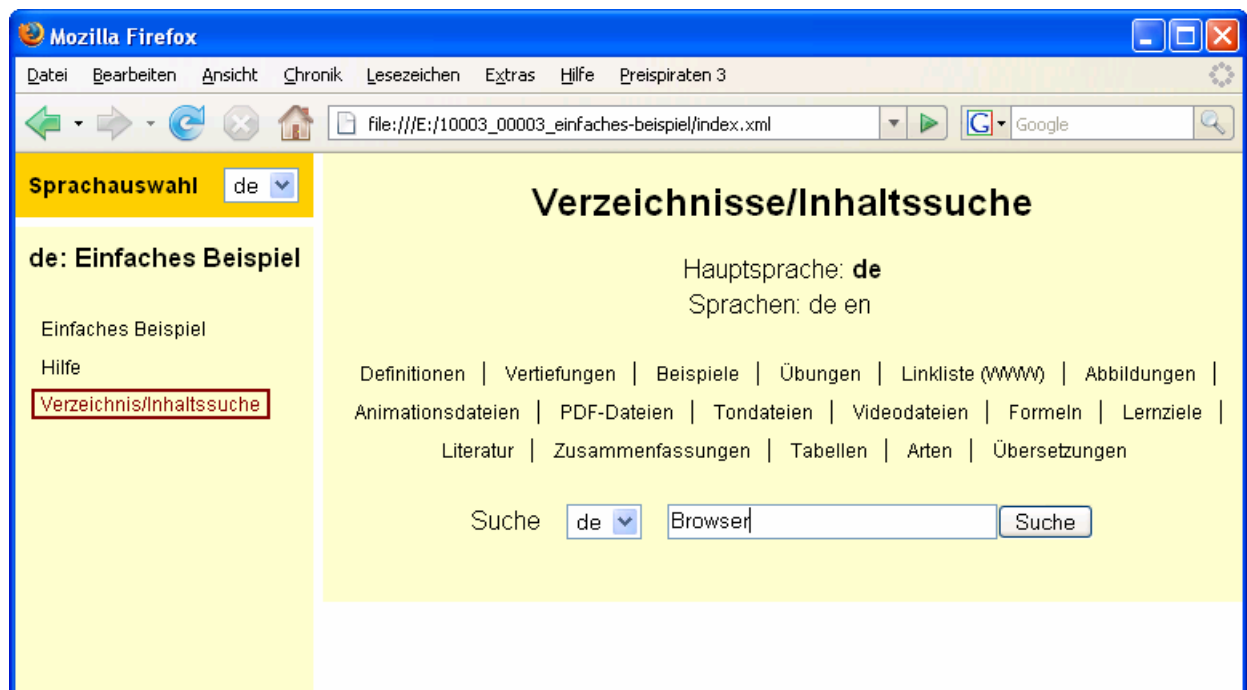


Abbildung 6: Inhaltssuche im WELPE-Framework (Volltext-Suchmaschine), Anzeige im Mozilla Firefox 2.0

Nachdem die Suche ausgelöst wurde, wird die JavaScript-Funktion xslt() (aus der xslt.js) mit Parameter-Werten aufgerufen. Als Parameterwerte werden u. a. die index\_search\_xml.xml, die index\_search\_html.xml und der Suchbegriff über Variablen übergeben (Abbildung 7).

```

//Der Suchbegriff wird in der Variablen searchString abgelegt
var searchString = document.formSearch.searchString.value;
...
xslt(...<xsl:value-of select="$indexSearchXmlXslt"/>','<xsl:value-of select="$indexSearchHtmlXslt"/>',
...,searchString);
...

```

Abbildung 7: Ausschnitt aus dem Quelltext der XSLT-Datei index\_sel.xml mit dem Aufruf der JavaScript-Funktion xslt()

Die aufgerufene JavaScript-Funktion xslt() (Abbildung 8) stößt die beiden XSLT-Transformationen (index\_search\_xml.xml, index\_search\_html.xml) an, der XSLT-Prozessor des Browsers führt sie durch und gibt das Ergebnis zurück. Da die Browser auf unterschiedliche Art und Weise XML und XSLT verarbeiten, muss über eine Browserweiche (in JavaScript umgesetzt) zwischen den aufrufenden Browsern unterschieden werden.

Weitere Hinweise zur JavaScript-Funktion xslt() sind im Ausschnitt des Quelltextes der xslt.js zu finden (Abbildung 8).

```

...
function xslt(...para_xsltFilename_1,para_xsltFilename_2,...para_searchString)
{
...
// Browserweiche fuer Netscape, Mozilla, Mozilla Firefox
if (document.implementation && document.implementation.createDocument)
{
...
// XML-Dokument erstellen und laden
xmlDoc = document.implementation.createDocument("", "", null);
...
xmlDoc.load(xmlUrl);
// Erstes XSLT-Dokument erstellen und laden
xsltDoc_1 = document.implementation.createDocument("", "", null);
...
xsltDoc_1.load(xsltUrl_1);
// XSLT-Processor-Instanz erstellen
xsltProc = new XSLTProcessor();
// XSLT-Datei in den vorher erstellten XSLT-Processor importieren
xsltProc.importStylesheet(xsltDoc_1);
// Parameter fuer XSLT-Datei setzen
xsltProc.setParameter(null, "searchString", searchString);
...
// Erste Transformation
firstTransformationResult = xsltProc.transformToDocument(xmlDoc);
...
// Zweite Transformation: Transformation des ersten Transformationsergebnisses
secondTransformationResult = xsltProc.transformToDocument(firstTransformationResult);

```

```

...
// Übergabe des zweiten Transformationsergebnisses (HTML) zur Anzeige an den Browser
...

```

Abbildung 8: Ausschnitt aus dem Quelltext der JavaScript-Datei xslt.js

Die `index_search_xml.xsl` wird von der in Abbildung 8 in Ausschnitten angegebenen JavaScript-Funktion `xslt()` aufgerufen und stellt sicher, dass alle XML-Dateien in der `imsmanifest.xml` geöffnet und nach dem Suchbegriff durchsucht werden. Die Ergebnisliste wird als XML-Input für die zweite XSLT-Transformation (`index_search_html.xsl`) an die JavaScript-Funktion `xslt()` (Abbildung 8) zurückgegeben.

```

<!-- Der Suchbegriff wird aus dem Formularfeld über den Aufruf der xslt() übergeben. -->
<xsl:param name="searchString"/>
...
<xsl:for-each select="document(' ../imsmanifest.xml',.)//imsmanifest:file">
  <xsl:if test="...(contains(@href,'.xml'))">
...
    <xsl:if test="(//@title | //text())[contains(.,$searchString)]">
      <xsl:call-template name="search">
...

```

Abbildung 9: Ausschnitt aus dem Quelltext der XSLT-Datei `index_search_xml.xsl`

Die `index_search_html.xsl` nimmt die XML-Ergebnisliste aus der ersten Transformation (`index_search_xml.xsl`) von der JavaScript-Funktion `xslt()` entgegen, entfernt doppelte Werte und sortiert sie. Anschließend übergibt sie das nach HTML transformierte Endergebnis (Abbildung 10) wieder zurück an die JavaScript-Funktion `xslt()`. Diese Funktion sorgt abschließend für die Anzeige im Browser (Abbildung 11).

```

...
<html>
...
<xsl:apply-templates select="/search/result...">
  <xsl:sort select="."/>
</xsl:apply-templates>
<xsl:template match="search/result">
...
  <a>
...
    <xsl:value-of select="."/>
  </a>
<!-- Die Pfadangabe zur Datei, in der sich der Suchbegriff befindet, wird unter dem Suchergebnis
angezeigt. -->
...
<xsl:value-of select="@pathSearchResult"/>
...

```

Abbildung 10: Ausschnitt aus dem Quelltext der XSLT-Datei `index_search_html.xsl`

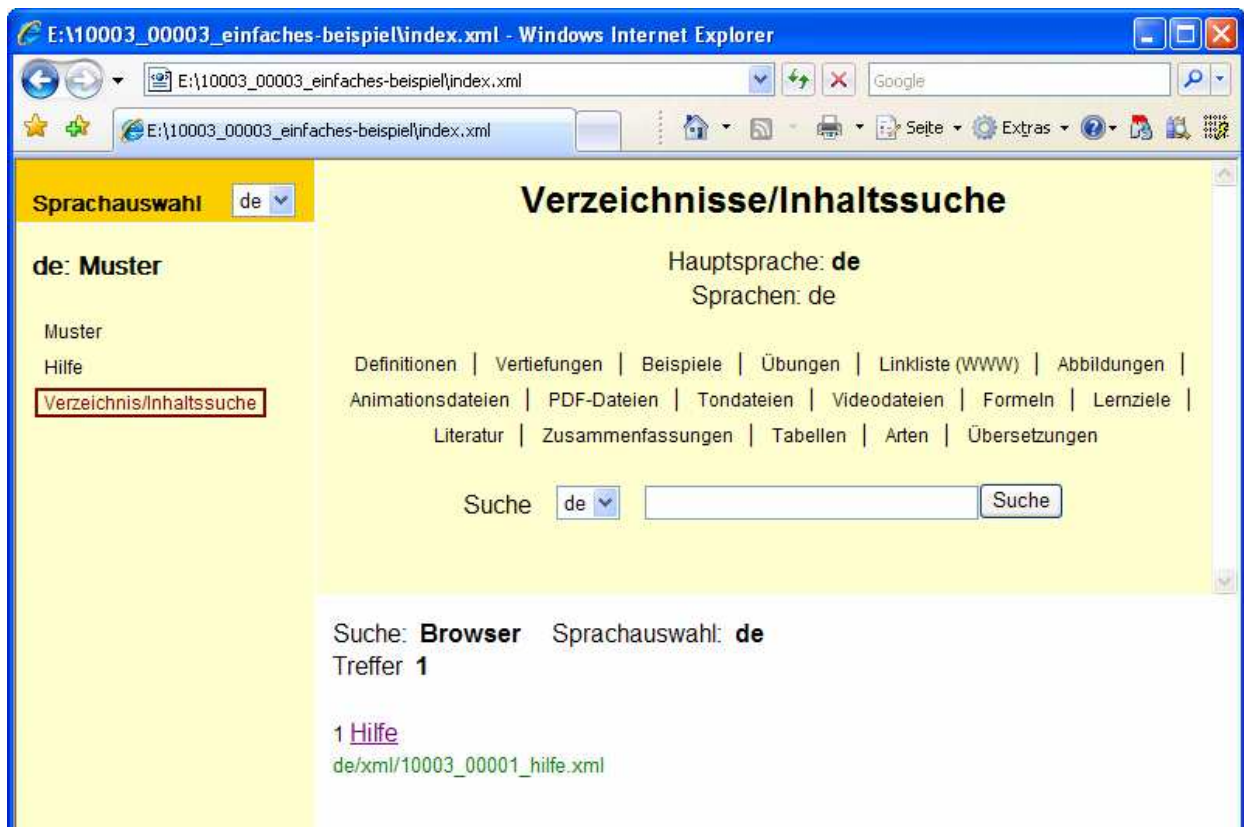


Abbildung 11: Ergebnis einer Volltextsuche im MS Internet Explorer 7

Klickt man auf ein Ergebnis, erscheint ein neues Browserfenster, in welchem der Suchbegriff an den verschiedenen vorkommenden Stellen markiert ist.

Die Umsetzung der Mehrsprachigkeit und die vollautomatische Erstellung von Verzeichnissen (z. B. Abbildungsverzeichnis, siehe auch Abbildung 11) wird im WELPE-Framework ebenfalls vor allem mit Hilfe von XSLT und JavaScript erreicht. Bei mehrsprachigen Inhalten wird für jede Sprache ein eigener Ordner innerhalb des WELPE-Frameworks angelegt. Die in verschiedene Sprachen übersetzten WLMML-Dateien werden in ihre spezifischen Sprach-Ordner abgelegt. In der imsmanifest.xml wird über Metadaten die Sprache der jeweiligen Lerneinheit festgehalten. Als Ergebnis kann z. B. die Sprache der Lerneinheit komplett umgestellt werden (siehe Abbildung 11 links oben) oder es können Absätze in den vorhandenen Sprachen übersetzt angezeigt werden.

## Diskussion und Ausblick

Neue Entwicklungen wie z. B. Asynchronous JavaScript and XML (AJAX, siehe GARRET 2005) und ECMAScript for XML (E4X) zeigen den Stellenwert von XML und JavaScript. Unter ECMA (2005) wird bezüglich ECMAScript for XML (E4X) festgehalten: "This Standard defines the syntax and semantics of ECMAScript for XML (E4X), a set of programming language extensions adding native XML support to ECMAScript...". JavaScript (ECMAScript) würde damit im Umgang mit XML um einige interessante Möglichkeiten erweitert. Inwieweit die Browser allerdings ECMAScript for

XML (E4X) umsetzen werden, bleibt abzuwarten. Zusätzlich muss darauf hingewiesen werden, dass JavaScript vom Benutzer deaktiviert werden kann.

Die im WELPE-Framework abgelegten Lerninhalte sind in den weltweit verbreiteten Browsern Microsoft Internet Explorer 6 und 7, sowie Mozilla Firefox 1.5 und 2 lauffähig. Laut ONESTAT (2006) halten die Browser beider Firmen zusammen einen Weltmarktanteil von über 95% (Stand Juli 2006). Die Unterstützung von JavaScript, XML und XSLT in den neuen Browser-Generationen Microsoft Internet Explorer 7 und Mozilla Firefox 2.0 lässt vermuten, dass diese Technologien auch weiterhin genutzt werden können.

Grundsätzlich sollte der konsequente Einsatz von Standards/Normen und Spezifikationen im WELPE-Framework eine zukunftssichere Verwendung und Austausch von Lerninhalten ermöglichen.

## Literaturverzeichnis

GARRET, J. J. (2005): Ajax: A New Approach to Web Applications, February 18, 2005, <http://www.adaptivepath.com/publications/essays/archives/000385.php>, Abruf am 2006-11-13

ECMA (1999): Standard ECMA-262, ECMAScript Language Specification, 3rd edition (December 1999), <http://www.ecma-international.org/publications/standards/Ecma-262.htm>, Abruf am 2006-09-13

ECMA (2005): Standard ECMA-357, ECMAScript for XML (E4X) Specification, 2nd edition (December 2005), <http://www.ecma-international.org/publications/standards/Ecma-357.htm>, Abruf am 2006-09-14

GAGARINOV, A., IVERSON, M. (2005): Transforming Word Documents into the XSL-FO Format, February 2005, [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odc\\_wd2003\\_ta/html/OfficeWordWordMLtoXSL-FO.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odc_wd2003_ta/html/OfficeWordWordMLtoXSL-FO.asp), Abruf am 2006-11-13

IEEE LTSC (2005): 1484.12.3, Draft 8, the final draft of the XML schema binding standard for LOM. 16 February 2005.

[http://ieeeltsc.org/wg12LOM/1484.12.3/Public/IEEE\\_1484\\_12\\_03\\_d8.pdf/view?searchterm=LOM](http://ieeeltsc.org/wg12LOM/1484.12.3/Public/IEEE_1484_12_03_d8.pdf/view?searchterm=LOM),  
Abruf am 2006-11-16

IMS CP (2003): IMS Content Packaging, [www.imsproject.org/content/packaging/index.cfm](http://www.imsproject.org/content/packaging/index.cfm), Abruf am 2003-06-15

LMML (2006): Learning Material Markup Language Framework LMML, <http://www.lmml.de/>, Abruf am 2006-09-25.

ONESTAT (2006): Global usage share Mozilla Firefox has increased according to OneStat.com, 09.06.2006, [http://www.onestat.com/html/aboutus\\_pressbox44-mozilla-firefox-has-slightly-increased.html](http://www.onestat.com/html/aboutus_pressbox44-mozilla-firefox-has-slightly-increased.html), Abruf am 2006-11-13

- PAAR, S. (2004): Der Einsatz von XSLT im Bereich E-Learning am Beispiel von IMS LD und LMML, Deutscher Verband Forstlicher Forschungsanstalten, Sektion Forstliche Biometrie und Informatik, 16. Tagung 2004 Freising, Die Grüne Reihe, herausgegeben von U. Wunn (Forschungsanstalt für Waldökologie und Forstwirtschaft Rheinland-Pfalz in Trippstadt) und H.-D. Quednau (TU München)
- SCORM (2006): Sharable Content Object Reference Model, <http://www.adlnet.gov/scorm/index.cfm> , Abruf am 2006-11-16
- STREHL, O., QUEDNAU, H. D., PAAR, S. (2005): Konzept einer standard-konformen E-Learning-Modul-Bibliothek, Deutscher Verband Forstlicher Forschungsanstalten, Sektion Forstliche Biometrie und Informatik, 17. Tagung 2005 Freiburg, Die Grüne Reihe, herausgegeben von U. Wunn (Forschungsanstalt für Waldökologie und Forstwirtschaft Rheinland-Pfalz in Trippstadt)
- SÜSS, C. (2000): Adaptive Knowledge Management: A Meta-Modelling Approach and its Binding to XML. In: H.-J. Klein (Ed.), 12.GI-Workshop Grundlagen von Datenbanken, Plön, TR 2005, Christian-Albrechts-Universität Kiel, Germany, 2000
- THROPP, S. (2004): The Impact of the Standardization Process on SCORM 2004, July 22, 2004. Advanced Distributed Learning (ADL). <http://www.adlnet.gov/scorm/articles/8.cfm> , Abruf am 2006-11-16
- W3C (1996): World Wide Web Consortium: Cascading Style Sheets, level 1, W3C Recommendation 17 Dec 1996, revised 11 Jan 1999, <http://www.w3.org/TR/CSS1>, Abruf am 2006-11-11.
- W3C (1999 a): World Wide Web Consortium: XSL Transformations (XSLT), Version 1.0, W3C Recommendation 16 November 1999, <http://www.w3.org/TR/1999/REC-xslt-19991116.html>, Abruf am 2004-02-11.
- W3C (1999 b): HTML 4.01 Specification, W3C Recommendation 24 December 1999, <http://www.w3.org/TR/html4/>, Abruf am 2006-11-16.
- W3C (2001): World Wide Web Consortium: Extensible Stylesheet Language (XSL), Version 1.0, XSL Formatting Objects (XSL-FO), W3C Recommendation 15 October 2001, <http://www.w3.org/TR/xsl/>, Abruf am 2004-02-11.
- W3C (2002): World Wide Web Consortium: XML Schema W3C Recommendation, <http://www.w3.org/XML/Schema>, Abruf am 2006-09-25
- W3C (2004): World Wide Web Consortium: Extensible Markup Language (XML) 1.1, <http://www.w3.org/TR/2004/REC-xml11-20040204/>, Abruf am 2004-02-20.
- WIKIPEDIA (2006): Begriffsbeschreibung Lernplattform, <http://de.wikipedia.org/wiki/Lernplattform>, Abruf am 2006-11-14.