XML mit PHP

Dr. Thomas Wieland
Siemens AG, Corporate Technology
XMLOne München 2001

s

Agenda

- PHP in fünf Minuten
- Möglichkeiten der XML-Verarbeitung in PHP
- Ereignisbasiertes Parsen mit Expat
- DOM-Aufbau mit libxml
- XSL-Transformation
- SOAP mit PHP

© T. Wieland, 2001. Alle Rechte vorbehalten

PHP in 5 Minuten

s

Was ist PHP?

- Ursprünglich ein einfaches Skript für eine „personal home page“
- Heute (PHP 4) eine leistungsfähige Skriptsprache für interaktive Webseiten mit Datenbankanbindung
- Rekursives Akronym: „PHP: Hypertext Processor“
- Open Source-Projekt mit eigener Lizenz und kommerziellem Support (Zend Technologies, Israel)
- Direkte Alternative zu ASP(+) und JSP/Servlets
- Teil der „**LAMP**“-Plattform
  - **L**inux + **A**pache + **M**ySQL + **P**HP

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Technik hinter PHP

- PHP-Skripts laufen auf dem Server ab
- PHP-Code kann von HTML umgeben sein
- Die Skripting-Engine kompiliert Skripts zu Bytecode, der dann interpretiert wird
- PHP läuft als Modul im Webserver
  - Je nach Servertyp auch als ISAPI- bzw. NSAPI-Filter oder als CGI-Anwendung

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Installation von PHP

- Download von www.php.net
  - als Quellcode oder Binärversion für Linux, Windows etc.
  - Oder bereits in der Linux-Distribution vorhanden (z.B. bei SuSE)
- Registrieren der Endung „.php“ beim Webserver
  - In httpd.conf (Apache), im Internet Services Manager (IIS) oder als MIME-Type (Netscape)

© T. Wieland, 2001. Alle Rechte vorbehalten

S

Syntax von PHP

- Die Syntax ist angelehnt an C, Perl und Java.
- Jedes Skript ist durch „<?php“ und „?>“ begrenzt.
- Variablen sind nicht typisiert
  - Sie müssen vor Verwendung nicht dimensioniert oder initialisiert werden.
  - Variablen werden mit einem vorangestellten $ gekennzeichnet

© T. Wieland, 2001. Alle Rechte vorbehalten

S

Hello World

```
<HTML><HEAD>
<TITLE>Hello PHP World</TITLE>
</HEAD><BODY>
<?php
    // "Hallo PHP!" ausgeben
    print "Hallo PHP!";
?>
</BODY></HTML>
```

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Weitere Programmelemente

- Bedingungen und Schleifen haben dieselbe Syntax wie in C, z.B.

```
for( $i = 0; $i <= 10; $i++ ) {
    // Rumpf der Schleife
}
```

- Funktionen werden ohne Parametertypen definiert
- Klassen mit Einfachvererbung sind ebenfalls möglich
  - Zugriff auf Klassenelemente über `$this`-Zeiger

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Umgang mit Variablen und Feldern

- Variablennamen können dynamisch vergeben werden:
  - `$a="Hallo"; $$a="Welt"` ⇔ `$Hallo="Welt";`
- Feldindizes werden automatisch hochgezählt:
  - `$a[]="Hallo"; $a[]="Welt";`
- Felder können auch mit Strings indiziert werden:
  - `$a["Hallo"]="Welt";`

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Verbindung zu externen Komponenten

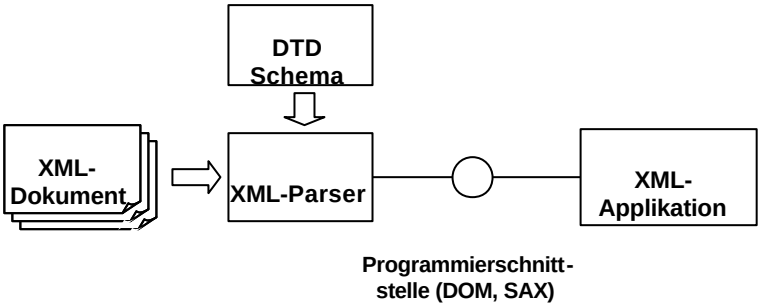
- PHP 4 enthält Unterstützung für Java, COM und .NET
  - Beans, Servlets, COM- sowie bereits .NET-Komponenten können lokal oder remote angesprochen werden
- Beispiel

```
<?php
  // Instanz der COM-Komponente anlegen
  $word = new COM( "word.application" ) or
            die( "Word konnte nicht geladen werden.");
  $word->Documents->Add();
  // ... (Weitere Befehle)
  $word->Quit();
?>
```

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Was braucht man für XML?



Aber: **DTD wird bald durch XSchema ersetzt (gleiche Funktion, aber selbst in XML)**

© T. Wieland, 2001. Alle Rechte vorbehalten

s

XML-Verarbeitung mit PHP

- Alle gängigen Techniken werden unterstützt
  - Ereignisbasiert (ähnlich zu SAX)
  - Dokumentbasiert (DOM)
  - XSL-Transformationen
  - SOAP-Kommunikation
- PHP kapselt jeweils leistungsfähige Open Source-Bibliotheken
- Daneben auch Zugriff auf Fremdkomponenten möglich (hier nicht betrachtet)
  - MSXML über COM
  - CLR über .NET
  - Xerces über Java

© T. Wieland, 2001. Alle Rechte vorbehalten

Ereignisbasierte XML-Verarbeitung in PHP

s

Die Expat-Bibliothek

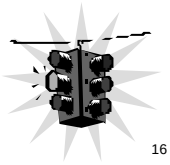
- Expat ist ein schneller, aber einfacher XML-Parser
  - nicht-validierend
  - ereignisorientiert
- Expat wird auch in anderen Projekten wie Apache, Mozilla und Perl genutzt
- Unterstützung für Expat ist in PHP 4 standardmäßig integriert und aktiviert
  - Expat muss daher i.A. nicht separat installiert werden
- PHP-Funktionen beginnen mit „`xml_`“

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Ereignisorientiertes Parsen

- Für jedes gefundene XML-Tag wird ein Ereignis ausgelöst
  - Öffnende und schließende Tags werden separat behandelt
- Auch andere XML-Bestandteile können als Ereignisse abgefangen werden
  - Z.B. Processing instructions oder String-Daten
- Der Parser analysiert nicht die XML-Baumstruktur!
  - Aufrufendes Programm muss sich den Weg durch das XML-Dokument selbst merken, sofern nötig



© T. Wieland, 2001. Alle Rechte vorbehalten

s

Programmbestandteile

1. Funktion zum Lesen der XML-Datei
   - Darin kann XML-Parser aufgerufen werden
2. Callback-Funktionen für Ereignisse
   - „Beginn eines XML-Elements gefunden“
   - „Ende eines XML-Elements gefunden“
   - „Datenbereich gefunden“
3. Evtl. Datenstruktur zum Speichern der Dokumenthierarchie

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Initialisierung des Parsers

```
// Erzeuge Instanz des XML-Parsers
$parser = xml_parser_create();

// Setze Parameter des XML-Parsers
xml_parser_set_option( $parser,
   XML_OPTION_CASE_FOLDING, true );

// Setze Handler für öffnende und schließende Tags
xml_set_element_handler( $parser, "start_element",
   "end_element" );

// Setze Handler für Daten
xml_set_character_data_handler( $parser, "daten" );
```

Daher alle Tags in Großbuchstaben

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Aufruf des Parsers

```
// Versuche, die Datei zu öffnen
if (! ( $fp = fopen("users.xml", "r" )))
  die( "Kann Datei nicht öffnen." );

// Lies Datei zeilenweise aus und parse die Zeile
while($data = fread( $fp, 4096 ))
  xml_parse($parser, $data, feof($fp));

// Gib XML-Parser frei
xml_parser_free($parser);
```

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Die Callback-Funktionen

```
// start_element() - wird vom XML-Parser bei öffnenden
// Tags aufgerufen
function start_element( $parser, $name, $attribute )
{
  // merke den aktuellen Tag
  global $curr_tag, $curr_user;
  $curr_tag = $name;
  if ($name == "USER")
    $curr_user = $attribute['ID'];
}

// end_element() - wird vom XML-Parser bei schließenden
// Tags aufgerufen
function end_element( $parser, $name )
{
  // hier keine Operation
}
```

© T. Wieland, 2001. Alle Rechte vorbehalten

s

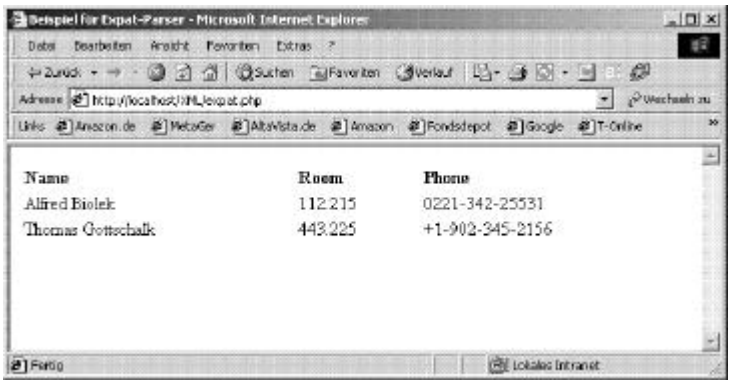
Callback-Funktion für Daten

```
// daten() - wird vom XML-Parser für CDATA-Bereiche
// aufgerufen
function daten( $parser, $data )
{
  // Mache globale Daten zugreifbar
  global $curr_tag, $curr_user, $users;

  // Merke den Inhalt der Daten
  switch($curr_tag) {
    case "USER": break;

    case "NAME": $users[$curr_user][0] = $data; break;

    case "ROOM": $users[$curr_user][1] = $data; break;

    case "PHONE": $users[$curr_user][2] = $data; break;
  }
}
```

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Demo



© T. Wieland, 2001. Alle Rechte vorbehalten

s

Alternative: Parsen in ein Array

- Der Code

```
$data = implode("", file("users.xml"));
xml_parse_into_struct($parser, $data, $arr, $i);
```

- speichert das Dokument in ein Array mit der Struktur

```
Array (
  [tag] => "Name des Tags"
  [type] => (open|close|complete)
  [value] => "Inhalt des Tags"
  [attributes] => Array (
                  [attrname] => "Name des Attr.„
                  .... )
  [level] => Hierarchiestufe
)
```

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Bewertung

Vorteile:

- Eventbasierte Schnittstelle sehr einfach und schnell
- Wenig Overhead an Datenstrukturen und Verwaltung

Nachteile:

- Hierarchie des XML-Dokuments kann nur schlecht verfolgt werden
- Code zum Parsen komplexerer Dokumente schnell aufgebläht

© T. Wieland, 2001. Alle Rechte vorbehalten

DOM-XML

s

DOM-Parsen mit *libxml*

- Seit PHP 4 wird auch das Parsen in einen Baum gemäß dem Document Object Model (DOM) unterstützt
- Verwendet wird dabei *libxml*, der Parser aus dem Gnome-Projekt
  - Kann validieren
  - Arbeitet baumorientiert
- Unterstützung für libxml muss installiert werden
  - Unter Linux durch Übersetzen von PHP mit der Option --with-dom
  - Unter Windows durch Kopieren der DLL und auskommentieren in php.ini
- Viele PHP-Funktionen beginnen mit „`domxml_`“

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Baumorientiertes Parsen

- Der Parser gibt Objekt der Klasse `Dom-Document` zurück
  - Repräsentiert das XML-Dokument als Baum-struktur
  - Methoden:
    - `root()` gibt die Wurzel zurück
    - `children()` die Unterknoten als Array
    - `dtd()` die DTD

- Abarbeitung z.B. durch rekursive Funktion, die für jeden Knoten des Baum aufgerufen wird

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Einlesen, Parsen und Auswerten

```
// Lies Datei zeilenweise aus
$data = implode("", file("users.xml"));

// Parse die Datei
if (!$doc = xmldoc($data)) {
  echo "Fehler beim Parsen.";
  exit;
}

// Durchlaufe rekursiv den Baum
process_node($doc->root());
```

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Unterscheidung nach Knotentyp

```
// Rekursive Funktion für den Durchlauf
function process_node($node)
{
  switch($node->type) {
    case XML_ELEMENT_NODE:
      $current_tag = strtoupper(trim($node->name));

      if ($current_tag == "USER")
        $curr_user = $node->getattr("id");
      break;

    case XML_TEXT_NODE:
      $data = $node->content;
```

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Nächste Rekursionstiefe

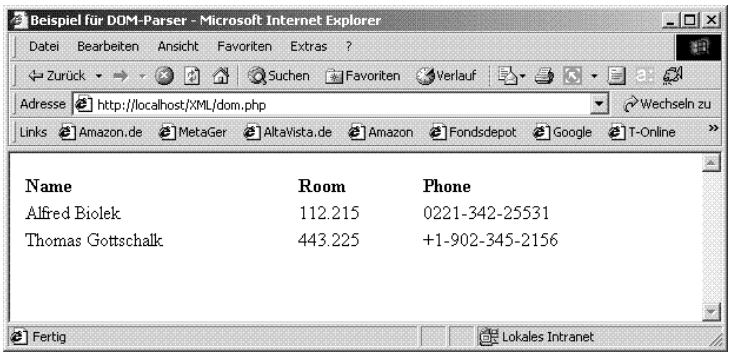
- Rufe Funktion mit Tochterknoten abermals auf, sofern welche vorhanden

```
$children = $node->children();

if (is_array($children)) {
  foreach($children as $child)
    process_node($child);
}
```

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Demo



© T. Wieland, 2001. Alle Rechte vorbehalten

s

Weitere Möglichkeiten

- Ausgabe eines kompletten Baumes

```
$tree = xmltree($xml);
print_r($tree);
```

- Aufbau eines neuen XML-Dokuments im Speicher

```
$doc = new_xmldoc("1.0");
$root = $doc->add_root("userlist");
$user = $root->new_child("user", "");
// ...
echo $doc->dumpmem();
```

© T. Wieland, 2001. Alle Rechte vorbehalten

S

Bewertung

Vorteile:

- Dokument als Ganzes verfügbar
- Leichtes Navigieren
- Hierarchieinformation im Speicher
- Schlanker Zugriffscode

Nachteile:

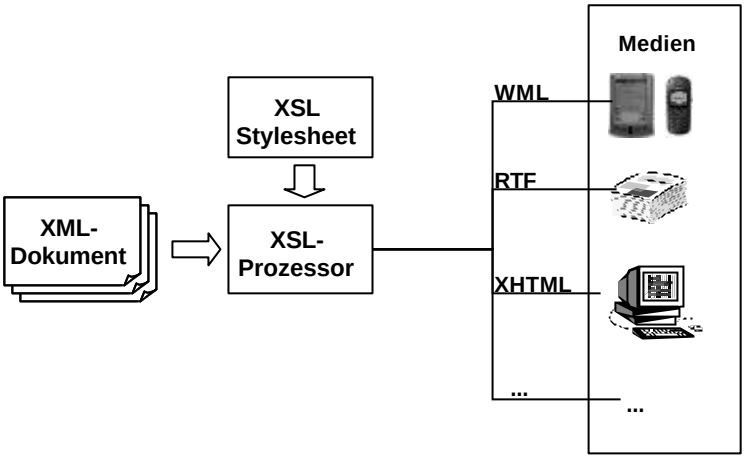
- Hoher Speicherbedarf, besonders bei komplexen Dokumenten
- Längere Verarbeitungsdauer als bei Expat

© T. Wieland, 2001. Alle Rechte vorbehalten

XSL-Transformation

S

Formatierung von XML-Dokumenten



© T. Wieland, 2001. Alle Rechte vorbehalten

S

XSL-Transformation aus PHP

- Verwendet wird der Sablotron XSL-Prozessor
  - Open-Source-Projekt mit MPL/GPL
  - Baut auf Expat auf
- Unterstützung für XSLT muss installiert werden
  - Unter Linux durch Übersetzen von PHP mit der Option --with-xslt
  - Unter Windows durch Kopieren von php_sablot.dll und auskommentieren in php.ini
- Die Funktionen beginnen mit „`xslt_`“

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Beispiel-Stylesheet

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" indent="yes" encoding="utf-8"
   />

<xsl:template match="/userlist">
  <body><table width="100%">
      <xsl:for-each select="user">
          <tr>
            <td><xsl:value-of select="name" /> </td>
            <td><xsl:value-of select="room" /> </td>
            <td><xsl:value-of select="phone" /></td>
          </tr>
        </xsl:for-each>
    </table></body></html>
</xsl:template>
</xsl:stylesheet>
```

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Transformation eines Dokuments

- Aufruf von xslt_transform() genügt

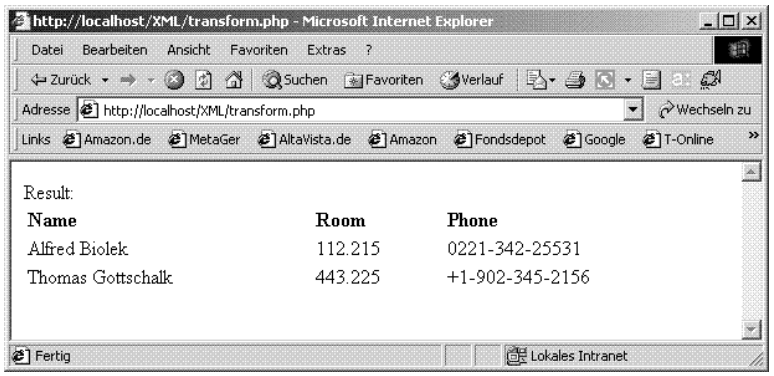
```
<?php
xslt_transform("users.xsl", "users.xml",
  "arg:/result", 0, 0, $result)
print "Result: " . $result;
?>
```

- Dateien werden im PHP-Include-Pfad gesucht
- Zusätzliche Argumente werden als assoziative Arrays übergeben

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Demo



© T. Wieland, 2001. Alle Rechte vorbehalten

s

Filtern aller Ausgaben

- Das XSL-Modul kann alle Ausgaben an die Standardausgabe mittels eines Stylesheets filtern:

```
xslt_output_begintransform("users.xsl");
$doc = new_xmldoc("1.0");
// ... $doc füllen
print $doc->dumpmem () // wird gefiltert
xslt_output_endtransform();
```

© T. Wieland, 2001. Alle Rechte vorbehalten

SOAP mit PHP

s

Das Middleware-Problem

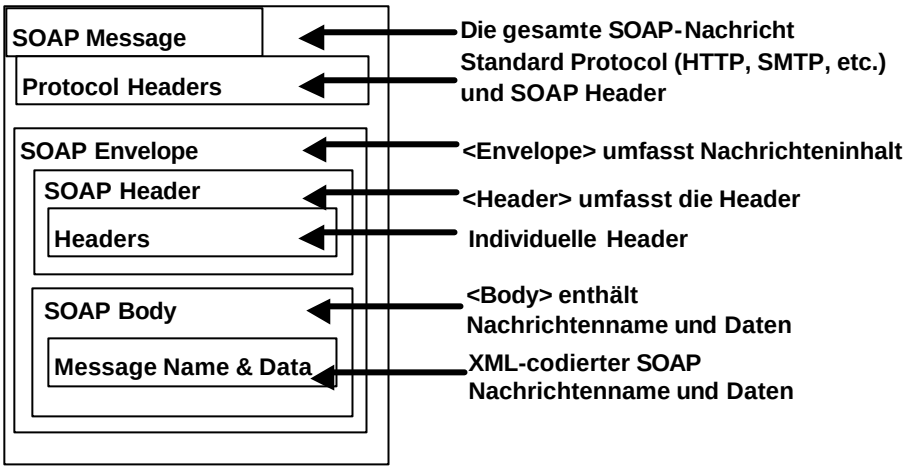
- Firewalls blockieren COM/IIOP-Anfragen, sind aber normalerweise für HTTP-Anfragen offen
  - ⇨ Setze auf HTTP als Transportprotokoll
- Wir müssen zudem ein Datenformat einführen, das selbst-beschreibend sein muss, um wirklich interoperabel zu sein.
  - ⇨ Definiere das Datenformat mit XML

Die Lösung: Simple Object Access Protocol SOAP
- Verlässt sich auf HTTP für die Kommunikation
- Selbstbeschreibendes einfaches XML-basiertes Nachrichtenformat
- Arbeitet zusammen mit:
  - Jedem Betriebssystem
  - Jeder Programmiersprache
  - Jeder Plattform

© T. Wieland, 2001. Alle Rechte vorbehalten

S

SOAP Nachrichten



© T. Wieland, 2001. Alle Rechte vorbehalten

S

Kommunikation mit SOAP

- SOAP definiert zwei Arten von Nachrichten: Anfragen und Antworten.
  - Anfragen enthalten das angesprochene Ziel sowie die in- und die in/out-Parameter.
  - Antworten umfassen einen Fehlercode oder das Resultat zusammen mit den in/out- und den out-Parametern.
- SOAP ist grundsätzlich nicht auf HTTP beschränkt, in der Verbindung mit PHP derzeit allerdings schon.
- SOAP erfindet nichts Neues, wo es bereits einen Standard gibt. Beispielsweise wird SSL bzw. HTTPS für Verschlüsselung benutzt.
- Durch die Verwendung von SOAP können Webserver so erweitert werden, dass sie das „Broker"-Architekturmuster voll unterstützen (so gen. Web Services).
- SOAP kann unterschiedliche Infrastrukturen überbrücken und so Anwendungen miteinander verbinden.
- SOAP ist ein W3C working draft (XML-Protocol).

© T. Wieland, 2001. Alle Rechte vorbehalten

s

SOAP mit PHP

- Erste prototypische Implementierungen verfügbar, u.a. von Gigaideas (China)
- Sowohl Clients als auch Server mit PHP möglich
  - Aber auch Interoperabilität z.B. mit Java
- Integration in PHP noch rudimentär
  - Keine Objekte als Parameter erlaubt, d.h. keine abstrakten Datentypen
  - Übertragung der Nachricht als „rohes“ XML, Parsen mittels Expat
  - Server-Methoden sind normale PHP-Funktionen, ohne besondere Auszeichnungen

© T. Wieland, 2001. Alle Rechte vorbehalten

s

SOAP-Client

```
$soap_msg=new soapmsg(
    'plus',         //Funktionsname
    'server.php',   //Implementierungsdatei
   array(          //Parameterliste
      new soapval($x,"int"), new soapval($y,"int")));

$client=new soap_client(
   "/XML/soap_php/rpcrouter.php", // Ort des Servers
   "localhost",            // Servername
   80);                    // Port

$return=$client->send($soap_msg);
$v=$return->value();
```

© T. Wieland, 2001. Alle Rechte vorbehalten

S

Demo

© T. Wieland, 2001. Alle Rechte vorbehalten

47

Zusammenfassung

s

Fazit

- PHP etabliert sich als schnelle und flexible Skriptsprache für Webapplikationen
  - Besonders zusammen mit Linux, Apache und MySQL (LAMP-Plattform)
- Die XML-Verarbeitung mit PHP ist sowohl ereignis- als auch baumorientiert möglich
- Durch den XSLT-Prozessor können auch XML-Ausgabe von PHP-Skripts verarbeitet werden
- SOAP ist mit PHP bereits in einfacher Form möglich
  - Interagiert auch mit anderen Plattformen, z.B. Java

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Links

- PHP-Home: www.php.net
- XML-Home: www.w3.org/XML/
- James Clarks expat: www.jclark.com/xml/
- XML-Bibliothek für Gnome: xmlsoft.org/
- DOM-XML-Funktionen: http://php.net/manual/en/ref.domxml.php
- XSLT-Prozessor Sablotron: www.gingerall.com/
- XSLT-Spezifikation: www.w3.org/TR/xslt
- PHP-SOAP: www.gigaideas.com.cn/phpsoap/
- Download der Beispiele: www.drwieland.de

© T. Wieland, 2001. Alle Rechte vorbehalten

s

Danke für die Aufmerksamkeit!

Fragen oder Anmerkungen?

© T. Wieland, 2001. Alle Rechte vorbehalten