METRIX
INTERNET DESIGN GMBH

Dynamische Websites mit XML und PHP4

Linuxtag 2002
07.06.2002, Karlsruhe

Stephan Schmidt

Inhalt der Session

- Über den Redner
- Strukturierung von Inhalten
- Transformation von XML
- Entwickeln einer XML-Struktur
- Einführung in patTemplate
- Installation und Architektur des patXMLRenderer
- Aufbau und Einsatz von Extensions
- Vergleichbare Applikationen
- Referenzen

Stephan Schmidt

- Web Application Developer bei Metrix Internet Design GmbH in Karlsruhe
- Nebenbei: LGPL Tools auf
  http://www.php-tools.de
- Entwickler von patXMLRenderer, patTemplate, patUser und anderen Tools

Strukturierung von Inhalten

- Klassische Webseiten in HTML:
  - » nicht strukturiert
  - » nicht von Maschinen lesbar
- Klassische Software-Entwicklung mit relationaler Datenbank:
  - » schwer von Menschen lesbar, da
    - – Abbildung der Struktur durch Relationen
    - – Verteilt auf viele Tabellen
    - – Kryptische IDs

Strukturierung mit XML

- Leicht von Menschen lesbar:
  - » Alle Daten in einer Datei
  - » Selbsterklärende Tagnamen
  - » Nähere Beschreibung durch Attribute
  - » Optische Gliederung durch Einrückung
- Leicht von Maschinen lesbar:
  - » Wohlgeformtes Dokument
  - » Reine Textdaten
  - » Validierung durch Schema oder DTD
- Trennung von Content und Layout

Beispiel für ein XML Dokument

```
<page>
    <headline>Dynamische Webseiten mit XML</headline>
    <section title="Über den Autor">
        <para>Der Autor hat nächste Woche Urlaub ☺</para>
    </section>
    <section title="Kurze Beschreibung">
        <para>Der Vortrag enthält</para>
        <list>
            <listelement>Beispiele</listelement>
            <listelement>ein wenig PHP Code</listelement>
        </list>
    </section>
</page>
```

Transformation von XML in HTML

Transformation jedes Tags in seine HTML Repräsentation, z.B.:

```
<headline>Dynamische Webseiten mit XML</headline>
```

...wird zu:

```
<font size="+1">
    <br><b> Dynamische Webseiten mit XML</b><br><br>
</font>
```

Transformation des ganzen Dokuments

- Für jeden Tag wird ein Template benötigt
- Richtige Reihenfolge bei Verschachtelungen (z.B. Listenelemente in einer Liste)

...wer macht das?

[XSLT]

Transformation mittels Stylesheet

und XSLT Prozessor

XSLT vs PHP4 und Templates

Transformation mit XSLT

- XSLT als Blackbox
- Keine Möglichkeit in der Transformationsprozess einzugreifen
- XSLT muss von Designer erstellt werden
- Offener Standard

Transformation mit PHP4 und Templates

- Volle Kontrolle über Transformationsprozess
- Leider kein offener Standard ☹

Transformation mit PHP4

- Ein Template für jeden XML Tag
  - XML Attribute werden zu Template-Variablen
  - Inhalt des Tags wird zur Template-Variable {CONTENT}

- Verwendung eines SAX-basierten Parsers (expat)
  - Simple Callback Methoden für Starttag, CData und Endtag
  - Rekursives Durchlaufen des Dokuments
  - Template wird beim Schließen des Tags geparst und an das Ergebnis angefügt

Entwickeln einer XML Struktur

- Aufteilen in logische Elemente z.B:
  - Navigation
  - Textparagraph
  - Liste
- Tag muss das Element beschreiben, z.B. <important> für wichtige Elemente in der Seite
- Keine reinen Layout Elemente, wie z.B.
- Festlegung der Seitenstruktur
  - XML Schema
  - DTD

Entwickeln einer XML Struktur 2

:: PATXMLRENDERER DEMO

» Home
» Einfache Beispiele
» Variablen
» Kontroll- Strukturen
» Datenbanken
» Metrix
» PHP Tools
» Kontakt

» Designwechsel
» XML-Quelle anzeigen

» Willkommen.

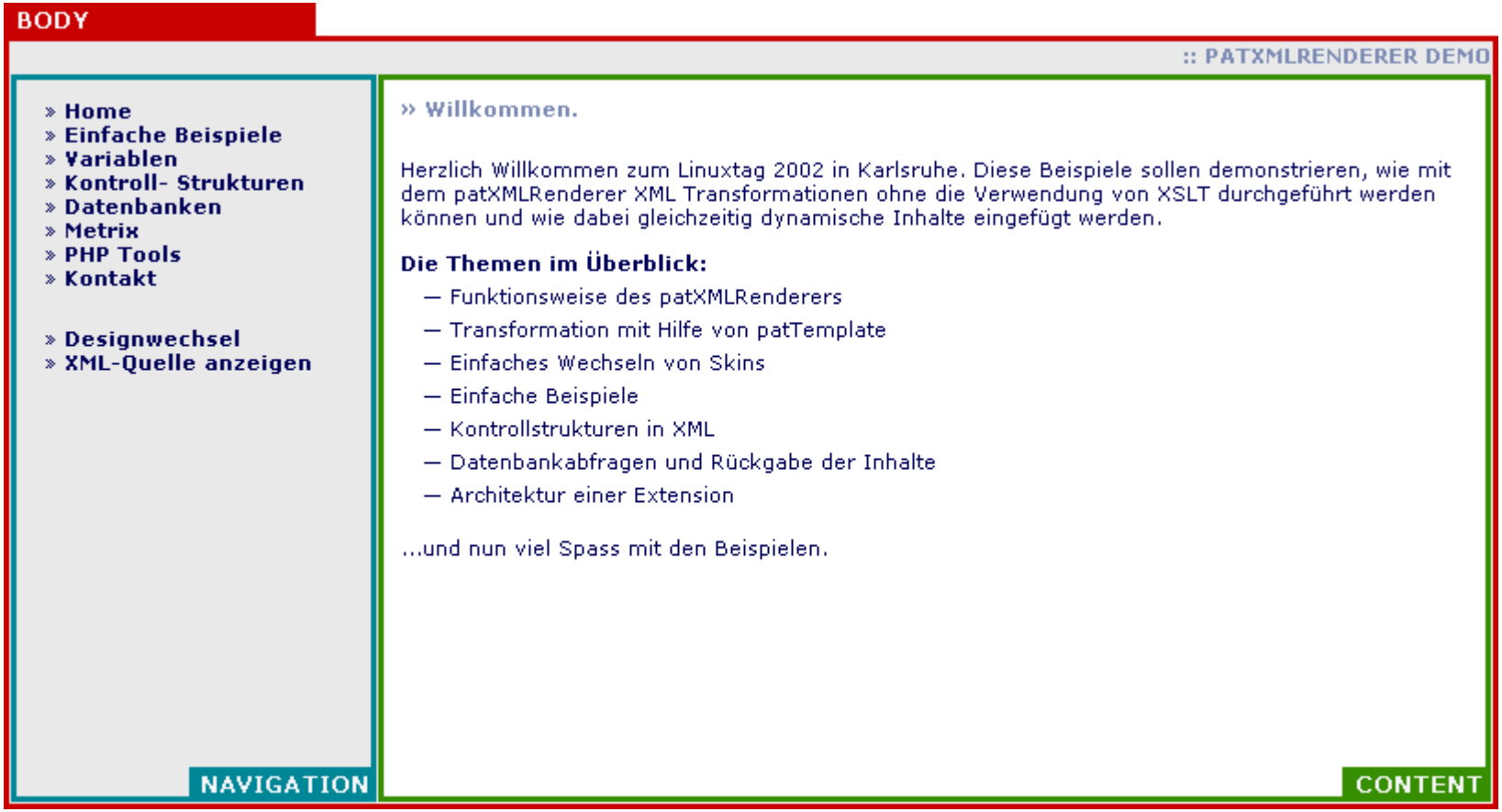
Herzlich Willkommen zum Linuxtag 2002 in Karlsruhe. Diese Beispiele sollen demonstrieren, wie mit dem patXMLRenderer XML Transformationen ohne die Verwendung von XSLT durchgeführt werden können und wie dabei gleichzeitig dynamische Inhalte eingefügt werden.

Die Themen im Überblick:

- Funktionsweise des patXMLRenderers
- Transformation mit Hilfe von patTemplate
- Einfaches Wechseln von Skins
- Einfache Beispiele
- Kontrollstrukturen in XML
- Datenbankabfragen und Rückgabe der Inhalte
- Architektur einer Extension

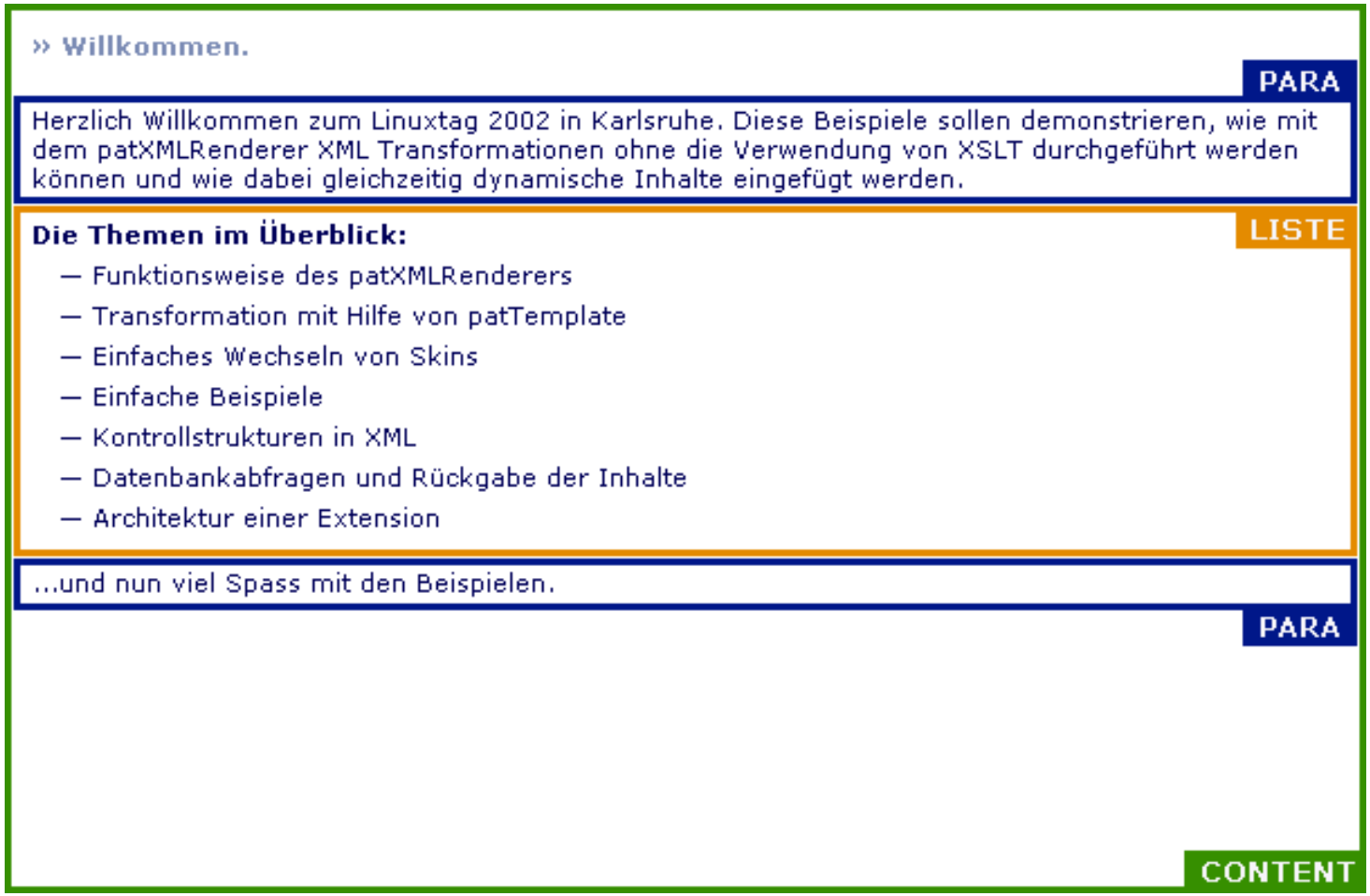
...und nun viel Spass mit den Beispielen.

Entwickeln einer XML Struktur 3



Entwickeln einer XML Struktur 4

Entwickeln einer XML Struktur 5



Einführung in patTemplate

- PHP Template Klasse unter LGPL
- Platzhalter für Variablen in Großbuchstaben und von { und } umschlossen
- Verwenden von <patTemplate:tmpl name="..."> Tags zum Markieren von Templates
- Weitere Eigenschaften der Templates werden als Attribute notiert, z.B: type="condition" oder whitespace="trim"
- Einfache switch/case und if/else Emulation durch <patTemplate:sub condition="..."> Tags

patTemplate Beispiel 1

Einfaches Template mit zwei Variablen, kann über den Namen "box" angesprochen werden.

```
<patTemplate:tmpl name="box">
<table border="1" cellpadding="5" cellspacing="0" width="{WIDTH}">
    <tr>
        <td>{CONTENT}</td>
    </tr>
</table>
</patTemplate:tmpl>
```

patTemplate Beispiel 2

Aufgabe:

Box soll in drei vorgegebenen Größen ausgegeben werden können: small, medium (default) und large

Lösung:

Condition Template zur Emulation von switch/case:

- Template-Typ ist "condition"
- Variable, die überprüft werden soll ist "size"
- drei mögliche Werte für "size": "small", "large" und "medium" (oder beliebiger anderer Wert)
  » drei Subtemplates.

patTemplate Beispiel 2

```
<patTemplate:tmpl name="box" type="condition" conditionvar="size">
    <patTemplate:sub condition="small">
        <table border="1" cellpadding="5" cellspacing="0" width="200">
            <tr><td>{CONTENT}</td></tr>
        </table>
    </patTemplate:sub>
    <patTemplate:sub condition="large">
        <table border="1" cellpadding="5" cellspacing="0" width="800">
            <tr><td>{CONTENT}</td></tr>
        </table>
    </patTemplate:sub>
    <patTemplate:sub condition="default">
        <table border="1" cellpadding="5" cellspacing="0" width="500">
            <tr><td>{CONTENT}</td></tr>
        </table>
    </patTemplate:sub>
</patTemplate:tmpl>
```

Installation patXMLRenderer

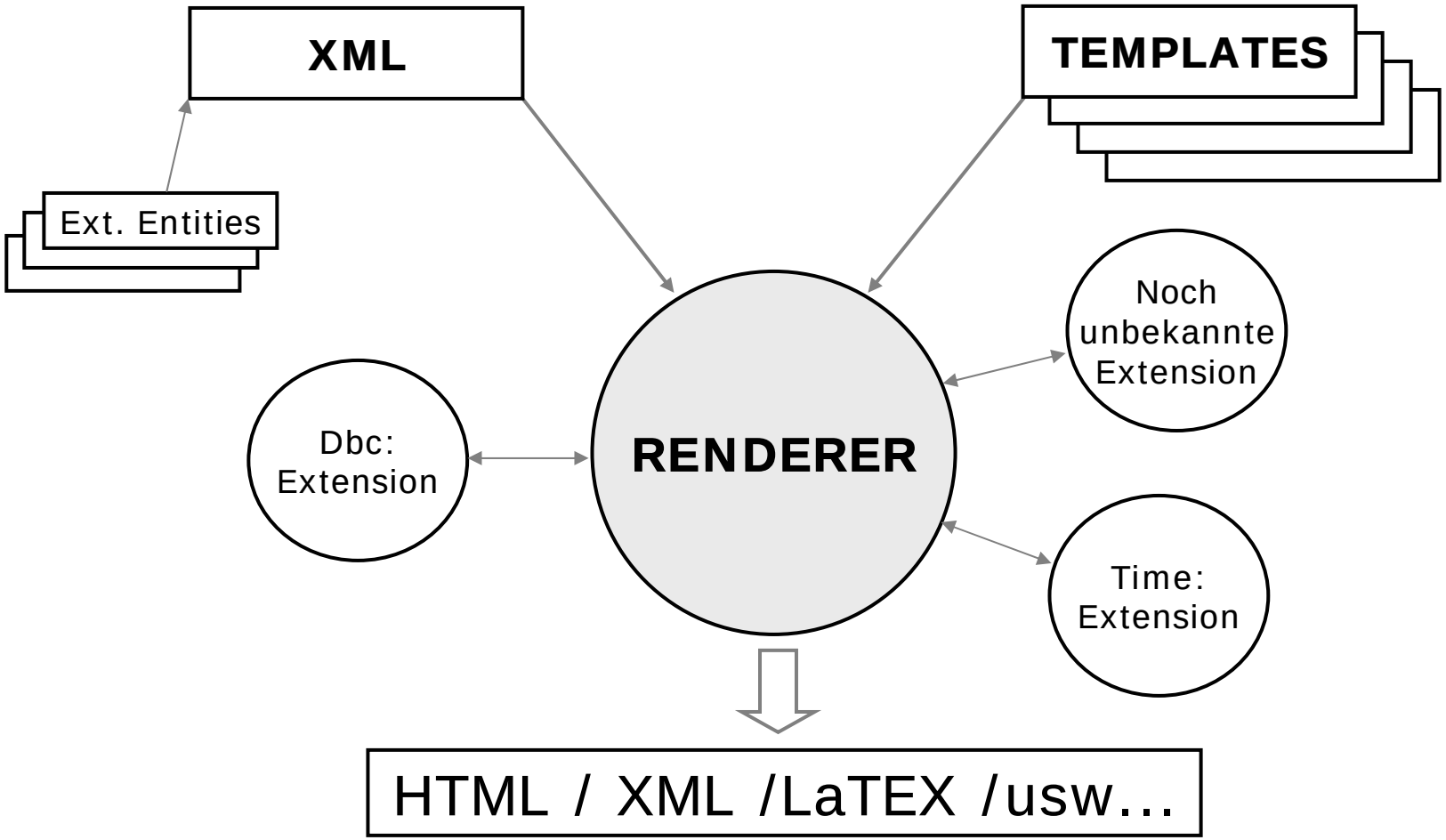
- Download von http://www.php-tools.de
- Archiv entpacken
- Anpassen der Pfade und sonstiger Optionen (cache, log, usw.) in der Konfigurationsdatei
- Erstellen der Templates
- Erstellen der XML Dateien mit dem Seiteninhalt

Fertig: » Kinderleicht! «

Arbeitsweise patXMLRenderer

- Dokument wird rekursiv geparst
- Tags ohne Namespace werden mit Template transformiert (Tagname = Templatename)
- Attribute stehen als Templatevariablen zur Verfügung
- Tags mit Namespace werden an Extensions (eigene Objekte) weitergegeben
- Extensions erzeugen Inhalt in Abhängigkeit der Attribute und des Contents, z.B. <time:current format="H:i"/>
- Rückgabe der Extension wird geparst, kann wieder Namespace Tags enthalten » Rekursion
- Externe Entitäten verhalten sich wie include()

Architektur



Struktur einer Extension

Jede Extension hat drei Handler:

- Starthandler legt Tags und Attribute auf einen Stack
- CData Handler speichert Daten in einer Variable
- End Element Handler führt switch/case Anweisung aus und gibt XML oder CData zurück.

Eigenschaften einer Extension

- Eindeutiger Name
- Versionsnummer
- Benötigte Version des patXMLRenderers
- Liste aller Tags mit der Angabe, ob der Tag gecached werden kann
- Liste aller Tags mit der Angabe, ob der Tag Markup (XML) zurück liefert

Durch Ableitung von der Basisklasse stehen Methoden zum Zugriff auf die Eigenschaften zur Verfügung

Bestehende Extensions

- Repository auf http://www.php-tools.de
- Automatische Dokumentation im Admin-Interface
- Beispiele für Extensions:
  - **<time:...>** Datums- und Uhrzeitfunktionen
  - **<dbc:...>** Datenbank-Schnittstelle
  - **<var:...>** Variablenzugriff
  - **<control:...>** Kontrollstrukturen
  - **<randy:...>** XML-API zum patXMLRenderer
  - **<file:...>** Dateioperationen
  - und weitere...

Zu trocken...?

...wie wär's mit Beispielen?

Vergleichbare Applikationen

Mindestens zwei weitere in PHP entwickelte Klassen:

- PEAR::XML_Transformer
  http://pear.php.net

- phpTagLib
  http://chocobot.d2g.com

Beide weitaus später als patXMLRenderer entstanden.
Beide nur schlanke Klassen für die Transformation, keine vollständigen Applikationen.

Vergleich

	patXMLRenderer	PEAR	phpTagLib
Templates	Ja, patTemplate	-	Ja, nur HTML
„Intelligente“ Templates	X	-	Nur mit PHP
Dynamischer Content	Ja, nur Objekte	Ja, Funktionen und Objekte	Vermischt mit Templates
Namespaces	X	X	-
Rekursion	Ja, beliebig	Ja, beliebig	-
External Entities	X	-	-
<?PHP ?>	X	-	X
Parameter	X	-	-
Administration	X	-	-
Ext. Repository	X	-	-

Referenzen

- PHP Application Tools
  (http://www.php-tools.de)
- XENA Website
  (http://xena.metrix.de)
- XENA Online Hilfe
  (Transformation in HTML und LaTex => PDF)

Ende

Vielen Dank für Ihre Aufmerksamkeit.

Weitere Informationen:

- http://www.php-tools.de
- schst@php-tools.de

Dank an:
Sebastian Mordziol, Gerd Schaufelberger, Stephan Eisler,
Metrix Internet Design GmbH