METRIX
INTERNET DESIGN GMBH

Dynamic transformations from XML to PDF-Documents with use of LaTeX

International PHP Conference 2003
Spring Edition
May 9th 2003, Amsterdam

Stephan Schmidt

Agenda

- About the speakers
- Types of documents
- Transforming XML documents
- Introduction to LaTeX
- Basic usage of LaTeX
- Converting LaTeX to PDF
- Dynamic creation of LaTeX and PDF documents
- Transforming XML documents
- Using patXMLRenderer to transform XML to PDF

Stephan Schmidt

- Web Application Developer at Metrix Internet Design GmbH in Karlsruhe/Germany
- Programming since 1988, PHP since 1998
- Publishing OS on http://www.php-tools.net
- Contributor to the German PHP Magazine
- Regular speaker at conferences
- Maintainer of patXMLRenderer, patTemplate, patUser and others

The problem

- Have been developing a really large application
- Writing technical as well as end-user documentation
- Documentation was available in XML (made available in the application as HTML)
- customers wanted documentation on paper

XML documents

- Readable by humans:
  - » self-explaining tag names
  - » self-explaining attribute names
  - » structured by indentation
- Readable by machines:
  - » Well-formed document
  - » only ASCII data
  - » Validation with DTD or schema
- Describe only the content

PDF documents

- Readable by humans:
  - » nice layout
  - » can be view on any platform
  - » can be easily printed
- Not readable by machines
  - » Binary document
  - » Mixture of content and layout
- Describe the content and layout

Getting the best of both

- Use XML documents for online documentation
- Use PDF for printed documentation

Disadvantage:

» two documents to maintain

Solution:

» create two documents from one source

Transforming XML

- Data is stored in an XML document
- Needed in different formats and environments
  - » Other XML formats (DocBook, SVG, ...)
  - » HTML
  - » Plain text
  - » LaTeX
  - » Anything else you can imagine
- Content remains the same

Transforming XML to HTML

Source document:

```
<example title=„My Example“>
  <greeting>
        Hello <imp>Clark Kent</imp>!
  <greeting>
</example>
```

Result of transformation to HTML:

```
<html>
    <head>
        <title>My Example</title>
    </head>
    <body>
        <h1>Hello <b>Clark Kent</b></h1>
    </body>
</html>
```

Transforming XML to PDF

- XML may only be transformed to ASCII documents
- PDF documents are binary files

Problem:

no direct transformation

Solution:

Step 1 » Transform XML to LaTeX
Step 2 » Transform LaTeX to PDF

Introduction to LaTeX

- based on TeX by Donald E. Knuth
- not a word-processor
- document preparation system for high-quality type-setting
- used for medium to large scientific documents
- can be used for any document: articles, books, letters, invoices, ...

Introduction to LaTeX (cont.)

- encourages you to concentrate on content instead of layout
- similar concept to XML, but not based on tags
- has to be "compiled" to view or print the result
- generates layout for your documents

Introduction to LaTeX (cont.)

- no WYSIWYG interface
- can be edited with your favourite editor (vi, emacs, HomeSite or even notepad, but **not** Frontpage)

  » can be created by any application or script that is able to create ASCII files.

```
<?PHP
    $fp = fopen( "file.txt", "w" );
    fputs( $fp, "Hello Clark Kent!" );
    fclose( $fp );
?>
```

Introduction to LaTeX (example)

```
\documentclass{article}
\title{Dynamic transformations of XML to PDF with LaTex}
\author{Stephan Schmidt}
\date{April 2003}

\begin{document}
  \maketitle
  We love XML, but everyone wants PDF.
\end{document}
```

Easy to understand

```
\documentclass{article}
```
» the document is an article

```
\title{Dynamic transformations of XML to PDF with LaTex}
```
» the title is "Dynamic transformations ..."

```
\author{Stephan Schmidt}
```
» Stephan Schmidt is the author

```
\date{April 2003}
```
» it has been written in April 2003

Easy to understand

```
\begin{document}
  \maketitle
  We love XML, but everyone wants PDF.
\end{document}
```

» document consists of a title (somehow generated) and some text.

LaTeX features

- Typesetting articles, technical reports, letters, books and slide presentations
- Control over large (and I really mean large) documents
- Control over sectioning, cross references, footnote, tables and figures
- Automatic creation of bibliographies and indexes
- Inclusion of images
- Using PostScript or Metafont fonts

Basic usage of LaTeX

LaTeX documents consist of:

- commands
  » text markup, paper definitions, etc.
- macros
  » collection of commands
- environments
  » split the document into logical components
- plain text
- comments

LaTeX commands

- start with a backslash ("\")
- parameters enclosed in curly braces ("{" and "}")
- optional parameters enclosed in brackets ("[" and "]") and separated by commas

Example:

```
\maketitle
\footnote{I am a footnote}
\documentclass[a4paper,twoside]{book}
```

LaTeX comments

- start with percent sign ("%")
- end at the end of the line

Example:

```
\documentclass{article}  % This will be an article
% This line is a comment and will be ignored later
```

LaTeX environments

- used to split the document into logical parts
- similar to tags in an XML document
- start with "\begin" command and end with "\end" command

Example:

```
\begin{document}
   % Place anything that is part of the document here
\end{document}
```

LaTeX special chars

- Some specialchars like "$", "{", "}", "_", etc. have to be quoted by adding a preceding backslash.
- "\\" marks the end of a paragraph
- "~" is similar to HTML's
- "\dots" will display "..."

Creating a document

- document always starts with "\documentclass" command to define the type of document
- responsible for the available commandset (no use for "\chapter" when you are writing a letter)
- used to define the basic layout style
- load packages after this command

Creating a document (cont.)

- include meta information ("\author", "\date", etc.) after the "\documentclass" command
- "\begin{document}" marks the start of the actual document (like <body> in HTML)
- Inside "document" environments any LaTeX command that structures the document may be used.

Creating a document (cont.)

```
\documentclass[a4paper,twocolumn]{article}
\usepackage{hyperref}

\title{Me and the myDocument}
\author{Me, of course}
\date{\today}

\begin{document}
\maketitle
\tableofcontents

\section{My relationship to Superman}
\subsection{How it started}
When I was twelve, Superman was my greatest hero.
\subsection{Our relationship grew stronger}
I first met him in person at the age of 16.
\subsection{Everything has to end}
When he died at the hands of {\em Doomsday}, I was really sad and devoted my life to Batman.
\section{My relationship to Batman}
My relationship to Batman started last week so there's not much to tell, yet.
But I already know some of his friends:
\begin{itemize}
  \item{\em Robin}, the Boy Wonder
  \item{\em Oracle}, the former Batgirl
\end{itemize}
\end{document}
```

Common LaTeX commands

- \section, \subsection and \subsubsection to structure the document
- \em to emphasize parts of the document
- \item to create lists
- \footnote (for footnotes, of course)
- \label, \bibitem, \ref and \href to create cross-references
- \includegraphics to include images
- \begin{table}, \begin{itemize} to create commonly used environments
- \tableofcontents, \listoftables and \listoffigures to create indexes
- ... and many more

Converting LaTeX to PDF

1. LaTeX needs to be installed on your system
   (Don't panic, installing LaTeX is mere child's play)

2. "latex myDocument.tex" creates "myDocument.dvi"
   (dvi means **D**e**v**ice **I**ndependent, can be converted to postscript, PDF or printer-native formats)

3. "xdvi myDocument.dvi" displays result

4. "dvipdf myDocument.dvi" creates "myDocument.pdf"

Converting LaTeX to PDF

If PDF is the only destination format, use

```
pdflatex myDocument.tex
```

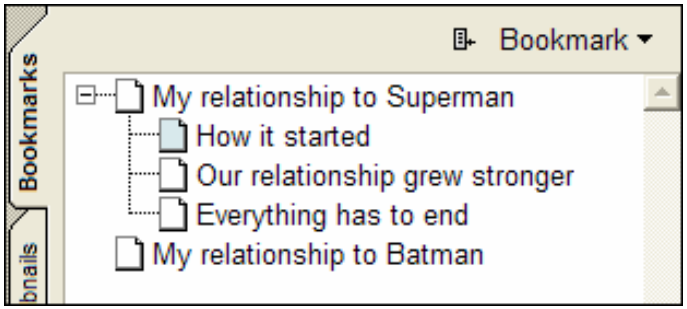
to generate a PDF file directly from your LaTeX source files.

Advantages:

- » faster
- » better support for fonts

Resulting document:

Bookmark table:



Inhaltsverzeichnis

1 My relationship to Superman

1.1 How it started

When I was twelve, Superman was my greatest hero.

1.2 Our relationship grew stronger

I first met him in person at the age of 16.

1.3 Everything has to end

When he died at the hands of *Doomsday*, I was really sad and devoted my life to Batman.

2 My relationship to Batman

My relationship to Batman started last week so there's not much to tell, yet. But I already know some of his friends:

- *Robin*, the Boy Wonder
- *Oracle*, the former Batgirl

Resulting files

After "pdflatex" has been called, several files are available in the folder:

- myDocument.pdf is the PDF file you wanted to create
- myDocument.log is a log file containing all log messages
- myDocument.toc contains the table of contents
- myDocument.out contains bookmarks for the PDF reader
- myDocument.aux contains all data needed for cross references

Two-pass transformations

- LaTeX parses file from top-down
- generates table of contents, anchor files for links, PDF bookmarks and stores them in external files
- This information often has to be included at the beginning of the document (e.g. table of contents)
- Latex file has to be parsed twice

  » two-pass transformation

  » pdflatex has to be called twice

Dynamic creation of LaTeX documents

LaTeX documents are plain text (like HTML)

» PHP can be embedded and any data inserted by using "echo"

```
\documentclass{article}
\begin{document}
  Hi, my name is <?PHP echo $_GET[“name”]; ?>.
\end{document}
```

Now open http://localhost/latex.php?name=Aquaman and save the result

» Your first dynamic LaTeX document!

Dynamic creation of LaTeX documents

But: No real automation :-(

as a delevoper needs to sit next to your webserver to handle all request » expensive!

Better:

Step 1 » Capture result with output control functions

Step 2 » Save result with file system functions

Step 3 » Transform file using system commands

dynamicLatex.php

```
<?PHP
    ob_start();
?>

\documentclass{article}
\begin{document}
  Hi, my name is <?PHP echo $_GET[“name”]; ?>.
\end{document}

<?PHP
  $latex = ob_get_contents();
  ob_end_clean();
  $fp = fopen( "dynamic.tex", "w" );
  fputs( $fp, $latex );
  fclose( $fp );
  system( "pdflatex dynamic.tex" );
  system( "shutdown -h now" );
?>
```

Not state-of-the-art

Creating larger and complex files can get messy:

- PHP and LaTeX commands in one file
- No separation of logic, content and layout

» you are a bad programmer!
(shame on you!)

State-of-the-art techniques

Implement the same techniques that are used in dynamic webpages:

- use templates
- store content in databases or XML
- use caching to gain performance

Transforming XML

- XSLT has been developed for the task of transforming XML documents
- XSLT stylesheets are XML documents
- Transforms XML trees that are stored in memory
- Uses XPath to access parts of a document
- Based on pattern matching
  ("When see you something that looks like this, do that...")
- Functional syntax

Sounds good?

» think again!

Drawbacks of XSLT

XSLT is domain specific:

- Developed to work with XML
- Creating plain text/LaTeX is quite hard, as whitespace is important (<xslt:text>)
- Transforming “world” to “W O R L D” is next to impossible

Drawbacks of XSLT

XSLT is verbose and circumstantial:

```
<xsl:choose>
  <xsl:when test="@author">
    <xsl:value-of select="@author"/>
    <xsl:text> says: </xsl:text>
    <xsl:value-of select="."/>
  </xsl:when>
  <xsl:otherwise>
    <xsl:text>Somebody says: </xsl:text>
    <xsl:value-of select="."/>
  </xsl:otherwise>
</xsl:choose>
```

Drawbacks of XSLT

XSLT is hard to learn:

- Functional programming language
- Complex structure (see “if/else” example”)
- XPath is needed
- Designer needs to learn it

Transforming XML using PHP

Transforming an XML document is easy:

1. Define a transformation rule for each tag
2. Start at the root element
3. Traverse the document recursively
4. Insert the transformation result to the parent tag
5. Go home early as you have completed the task faster than with XSLT.

Creating transformation rules

Rules are simple:

- “When you see this, replace it with that.”
- Implemented in PHP using templates
- Attributes of the tag are used as template variables
- PCData of the tag is used as template variable “CONTENT”

Example

XML

```
<section title="XML and PDF">
    <para>We love XML, but everybody wants PDF.</para>
</section>
```

Template for <section>

```
<table border="0" cellpadding="0" cellspacing="2" width="500">
        <tr><td><b>{TITLE}</b></td><tr>
        <tr><td>{CONTENT}</td></tr>
</table>
```

Template for <para>

```
<font face="Arial" size="2">{CONTENT}<br></font>
```

Example (Result)

```
<table border="0" cellpadding="0" cellspacing="2"
   width="500">
   <tr><td><b>XML and PDF</b></td><tr>
   <tr><td>
        <font face="Arial" size="2">
        We love XML but, everybody wants PDF.<br>
        </font>
   </td></tr>
</table>
```

Don’t reinvent the wheel

There already are XML transformers available for PHP:

- patXMLRenderer
  http://www.php-tools.net
- PEAR::XML_Transformer
  http://pear.php.net
- phpTagLib
  http://chocobot.d2g.com

Installation of patXMLRenderer

- Download archive at http://www.php-tools.de
- Unzip the archive
- Adjust all path names and options in the config file (cache, log, etc.)
- Create the templates (transformation rules)
- Create your XML files
- Let patXMLRenderer transform the files

Finished: » It's mere child's play «

Introduction to patTemplate

- PHP templating class published under LGPL
- Supports LaTeX templates when instantiated with `$tmpl = new patTemplate( "tex" );`
- Placeholder for variables have to be UPPERCASE and enclosed in { and } or <{ and }> if used with LaTeX templates
- Uses <patTemplate:tmpl name="..."> tags to split files into template blocks that may be addressed seperately
- Other Properties of the templates are written down as attributes, e.g: type="condition" or whitespace="trim"
- Emulation of simple switch/case and if/else statement by using <patTemplate:sub condition="..."> tags

patTemplate Example

simple Template with two variables

(Corresponds to the XML tag <box>)

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

patTemplate Example 2

Task:

Box should be available in three sizes: “small”, “large” and “medium” (default)

Solution:

Condition Template to emulate a switch/case statment:

- Template type is "condition"
- Variable that should be checked is called "size"
- Three possible values for "size": "small", "large" and "medium" (or any other unknown value)

  » three Subtemplates.

patTemplate Example 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>
```

Transforming XML to LaTeX

```
<?xml version="1.0" standalone="yes"?>
<article title="Me and the superheroes, part 2">
  <paragraph title="I lied to you">
    When I was talking about <imp>Superman</imp>,
    lied. He came back from the dead and rose to the glory
    he once had.
  </paragraph>
</article>
```

The LaTeX template

```
<patTemplate:tmpl name="article">
\documentclass[a4paper,twocolumn]{article}
\usepackage{hyperref}

\title{<{TITLE}>}
\author{Me, of course}

\begin{document}
\tableofcontents
<{CONTENT}>
\end{document}
</patTemplate:tmpl>

<patTemplate name="paragraph">
\section{<{TITLE}>}
<{CONTENT}>\\
</patTemplate:tmpl>

<patTemplate:tmpl name="imp" whitespace="trim">
{\em <{CONTENT}>}
</patTemplate:tmpl>
```

The Result

```
\documentclass[a4paper,twocolumn]{article}
\usepackage{hyperref}

\title{Me and the superheroes, part 2}
\author{Me, of course}

\begin{document}
\tableofcontents
\section{I lied to you}
When I was talking about {\em Superman}, I lied. He came back
from the dead and rose to the glory he once had.\\
\end{document}
```

Creating the PDF document

What are you waiting for?

Step 1 » Save the resulting LaTeX document

Step 2 » Use system("pdflatex myDocument.tex") to create a PDF document

Step 3 » Use header("Location: myDocument.pdf") to start the download.

To infinity ... and beyond!

patXMLRenderer can do even more for you:

- Supports overloading of namespaces to include any dynamic content (files, databases, rss streams, etc).
- Caching mechanism
- Logging
- Administration interface
- Offline generation of plain HTML

Simple Example

```
<example>
    Today is <time:current format="m-d-Y"/>.
</example>
```

Will be transformed to...

```
<example>
    Today is 05-09-2004.
</example>
```

...Which will then be transformed to LaTeX using the rules defined in the templates.

patXMLRenderer Example

```
<page>
  <dbc:connection name="foo">
    <dbc:type>mysql</dbc:type>
    <dbc:host>localhost</dbc:host>
    <dbc:db>myDb</dbc:db>
    <dbc:user>me</dbc:user>
    <dbc:pass>secret</dbc:pass>
  </dbc:connection>
        ...place any XML code here...
  <dbc:query connection="foo" returntype="assoc">
    SELECT id,name,email FROM authors
    WHERE id=<var:get scope="_GET" var="uid"/>
  </dbc:query>
<page>
```

patXMLRenderer Example (Result)

```
<page>
        ...any static XML...
  <result>
    <row>
      <id>1</id>
      <name>Stephan</name>
      <email>schst@php-tools.de</email>
    </row>
  </result>
</page>
```

Existing Extensions

- Repository on http://www.php-tools.net
- Examples:
  - » **<xml:...>** for XML syntax highlighting
  - » **<php:...>** for PHP syntax highlighting
  - » **<dbc:...>** database interface
  - » **<var:...>** access to variables
  - » **<control:...>** control structures
  - » **<rss:...>** to include content from RSS feeds
  - » **<file:...>** file operations
  - » and many more...
- Allow you to develop "XML Applications"

The End

Thank you!

More information:

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

Thanks to:

Sebastian Mordziol, gERD Schaufelberger, Metrix Internet Design GmbH