Getting started with plasTEX

Tim Arnold

Abstract

The software package plasTEX converts LATEX documents to other markup languages. This article describes typical usage with examples of how to create HTML and DocBook XML from LATEX sources, an overview of how to extend the package to handle custom commands and environments, and a demonstration of converting a simple LATEX source file.

1 Introduction

1.1 What plasTEX does

plasTEX is a document-processing framework that converts LATEX sources to HTML, XML, and other markup formats. The author uses plasTEX as part of a publishing workflow to produce statistical documentation at SAS Institute Inc.

The software is released as free and open source, under the MIT license. It is written in Python by Kevin Smith of SAS.

1.2 Math support

plasTEX is designed to use plug-ins for math support. You select the format of the math output by specifying the plug-in that plasTEX should use to render the document.

This plug-in design keeps plasTEX flexible so it can produce a mathematical format that is most appropriate for the current state of browser capabilities. Once browsers fully support MathML and their display is production quality, any software program that produces MathML from LATEX math can be used as the engine.

The default plug-in is `dvipng`, which replaces mathematics with images in the form of PNG formatted bitmap graphics. Optionally, you can use the `dvisvg` plug-in, which produces mathematics in SVG vector graphics.

2 How it works

2.1 Interface

plasTEX provides a command line interface. The interface includes options that enable you to:

- specify themes and navigational elements. A theme is a special template that sets the look and feel of the header, footer, and navigation elements such as breadcrumbs. plasTEX comes with several themes. You can specify a particular theme to match each output format (for example, HTMLHelp, EclipseHelp, and JavaHelp).
- specify input and output encoding. Different input encoding provides options for writers. Different output encoding enables matching the output to the destination; for example, HTMLHelp must be encoded as Windows-1252.
- set specific counters. It is useful to be able to process extremely large books one chapter at a time. Using the command line option, you can set the chapter counter for each chapter.
- set the depth of the table of contents and section numbering.
- specify image generation engine. This allows for PNG or SVG format mathematics and enables flexibility to use any future external process for handling LATEX math.

2.2 Internals

There are two steps in plasTEX processing:

1. plasTEX parses the LATEX source to create a *document object model* (DOM), which is a nested, tree-like data structure that contains the content plus the document data elements for the commands, environments, and their arguments.
2. plasTEX renders the DOM to the output format by combining a set of templates with the document data.

We now describe these steps in more detail.

2.2.1 Create the document object model

For every built-in LATEX command and environment, plasTEX provides a corresponding Python class so it can recognize and digest the tokens as they are encountered in a document.

As an example, plasTEX implements the LATEX `\framebox` command:

```
\framebox[width][position]{text}
```

with the following snippet of Python code, which defines the `framebox` class:

```
class framebox(TextBoxCommand):
    args = '[ width ] [ pos ] self'
```

The `framebox` class inherits from the internal plasTEX base class `TextBoxCommand` and maps the arguments to variables. During parsing, plasTEX reads the `framebox` command according to its definition and saves the resulting named tokens (values of the variables) in the document data structure. These variables are used in the next step, rendering.

2.2.2 Render the document

plasTEX rendering uses templates (a popular methodology in web publishing frameworks). A data structure (DOM) and a set of templates are combined to automatically generate documents.

Suppose you have a `framebox` command in your document:

```
\framebox[15em]{Note: this is an example}
```

LaTeX renders the markup as follows:

Note: this is an example

When plasTeX encounters the markup, it parses the `\framebox` into its constituent data and finally renders that data to HTML with the following template:

```
name: framebox
<span tal:attributes="style
  string:width:${self/attributes/width}"
  tal:content="self"></span>
```

The final output appears as follows:

```
<span style="width:15em">
Note: this is an example</span>
```

2.2.3 Summary of plasTeX processing

plasTeX parses each of the LaTeX commands and text in the source document to create the DOM. Parsing occurs one time for each input document.

The plasTeX parser must recognize every command and environment encountered in the document. plasTeX understands built-in LaTeX and TeX commands in addition to the following packages:

a4wide	alltt	amsart	amsfonts
amsmath	amssymb	amsthm	babel
beamer	changebar	color	comment
fancybox	fancyhdr	fancyvrb	float
fontenc	geometry	graphics	graphicx
hyperref	ifpdf	ifthen	inputenc
keyval	lipsum	longtable	makeidx
minitoc	natbib	pslatex	rotating
shortvrb	subfig	subfigure	textcomp
times	ucs	url	verbatim
wrapfig			

To create the output, plasTeX applies a set of corresponding templates for each document element that it encountered. The rendering step can be done multiple times using different sets of templates to create different output formats from a single DOM.

2.3 Rendering formats

plasTeX is bundled with the following template sets, i.e., it can render documents into these formats:

- HTML
- well-formed XML (the internal representation of a document within plasTeX is well-formed XML, which uses the plasTeX namespace)
- DocBook 4.5 and 5.0 XML
- plain text (useful to "detex" documents since all tagging is removed except for math)
- ePub, an emerging e-book standard format based on XHTML
- S5, a simple standards-based slide show system
- BrlTeX, an open source LaTeX-to-braille translator that is designed to handle math

2.4 Extending plasTeX

plasTeX can be extended in two ways:

- You can add new commands or environments that plasTeX can understand and parse by creating a corresponding Python class. The new class enables the plasTeX parser to tokenize documents that contain the new markup. This can often be a simple subclass of an existing class.
- You can add a new type of output format by adding a set of corresponding templates so that the plasTeX renderer can produce the appropriate output for each element.

3 Examples & demonstrations

The presentation of this paper included a live demonstration of converting LaTeX documents. To view the demonstration, install the plasTeX distribution and give the appropriate command, as follows:

- `plastex sample2e.tex`
  renders the familiar `sample2e.tex` example file into HTML. You can view the contents with any browser; open the file `sample2e/index.html`. See Figure 1.
- `plastex --renderer Text \`
  `--split-level 0 sample2e.tex`
  renders the same source file into a single file of plain text, leaving the math as it was entered in the document.

Also during the presentation, two research articles in LaTeX format were downloaded from the website `ArXiv.org` and rendered to HTML. (Two small bugs were noted by audience members. The bugs have been fixed in the current version of the plasTeX distribution.)

As a larger example, the *Python Library Reference* (a substantial documentation package written in LaTeX) was converted to HTML with no errors in about 45 minutes time.

In short, give plasTeX a try on your favorite documents; it can handle a lot.

4 Summary

In conclusion, plasTeX provides an easy-to-use command line interface with which to convert LaTeX sources to a variety of output formats. The framework can easily be extended by adding Python classes or templates, making it a useful tool for both simple documents as well as production-quality publishing workflows.

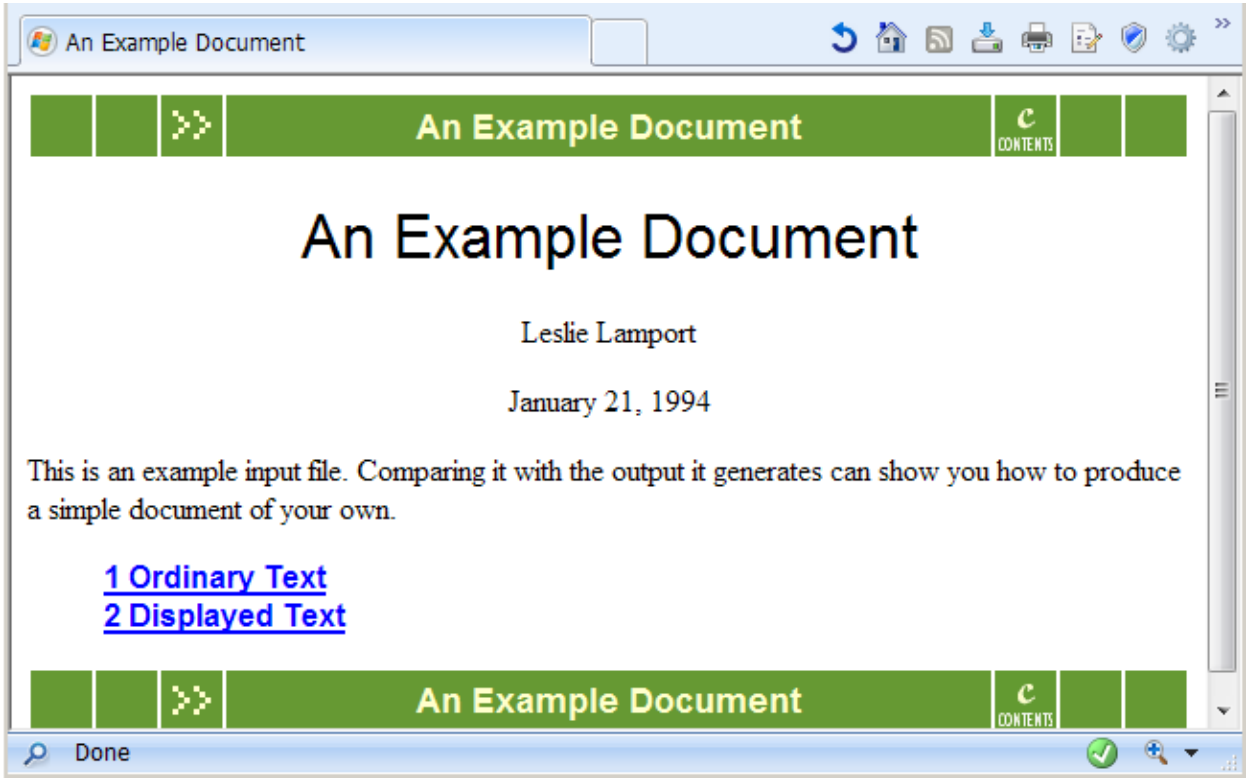

Figure 1: PlasTEX HTML output for `sample2e.tex`.

5 References

ArXiV is an e-print service in the fields of physics, mathematics, nonlinear science, computer science, quantitative biology and statistics.
`http://arxiv.org/`

BrlTEX is an open source LATEX-to-braille translator.
`http://brltex.sourceforge.net/`

DocBook is a schema maintained by the DocBook Technical Committee of OASIS. It is particularly well suited to books and papers about computer hardware and software.
`http://www.docbook.org/`

IDPF (International Digital Publishing Forum) is a trade and standards organization dedicated to the development and promotion of electronic publishing.
`http://www.idpf.org/`

plasTEX is a LATEX document-processing framework written entirely in Python, with the capability to output in many formats.
`http://plastex.sourceforge.net/`

S5 is a slide show format based entirely on XHTML, CSS, and JavaScript.
`http://www.s5.com/`

◇ Tim Arnold
SAS Institute Inc.
Cary, NC USA
`tim dot arnold (at) sas dot com`