How To Write A Minimal LaTeXML Binding

Author: Hang Yuan, Jinbo Zhang

Supervisor: Michael Kohlhase

February 15, 2015

Chapter 1

Introduction

LaTeX has been widely used as the word processing tool among scholars, especially when one needs to use large quantities of mathematical representations. LaTeXis also good for those who are meticulous about typographical quality of the documents. However, LaTeX lacks of conversion tool to XML which Digital Library of Mathematical Functions uses for delivery. DLMF developed LaTeXML , trying to make a new typesetting system that allows users to be able to focus more on the content, but not the style, by providing extensive ways of customizations. In order to achieve this goal, building up the document class binding seems crucial, and yet LaTeXML seems fairly unfathomable for beginners. We want to make it easier for those who want to pick up using LaTeXML in the future, by going through how to construct a minimal LaTeXML binding step by step.

This document does not cover advanced topics related LaTeXML, and thus if you are interested in understanding how this and how that work, very likely the LaTeXML manual will serve your needs better.

Chapter 2

Understanding LaTeXML

2.1 Using LaTeXML

The first thing we want to talk about is, what aspects of LaTeXML we are going to cover, and then we will move onto the workflow of creating your first LaTeX document class binding. In this tutorial, we use command:

latexml for converting TeX document into *.XML

The general command for conversion is

latexml options –destination=doc.xml doc

Or simply you only supply with the TeX file and the result will be standard output which is totally fine as well, based on your needs. One quick note here about LaTeXML installation, when you think you have finished installing LaTeXML, run a simple conversion command within mockDoc.tex's directory. You should be able to see an XML interpretation of mockDoc.tex either in a form a standard output or a newly-generated XML file. If you have something that differs from the expected and you have already checked your LaTeXML package multiple times, maybe you have overlooked some prerequisites such as libxml2 and libxslt.

Note: Now in order to make better use of our document class binding in the future, we need to know how LaTeXML operates in

different stages. The stages are like the following: Digestion , construction, rewriting, math parsing and serialization. However we are not going to discuss this in detail here. For the interested users, the LaTeXML manual is a good source.

2.2 LaTeXML Binding

The conversion from TeX to XML is processed by LaTeXML. Basically LaTeXML maps the TeX makeup to the XML markup, more specifically macros, primitives and constructors. That's why you are able to customize the conversion between TeX and XML, in three ways, modifying the bindings used by **latexml**, adding your own bindings that has not been implemented, and even creating your own TeX style and LaTeX binding which is exactly the goal of this tutorial.

2.2.1 Minimal LaTeXML Structure

Since LaTeXbinding is essentially a Perl module, we need to initialize a binding file by add the followings in the beginning of the document class:

package LaTeXML::Package::Pool; //load text package
use strict; // catch errors and stop when encounter one
use LaTeXML::Package;

//your customization here

use warnings; // give warnings
1; //make sure Perl work properly

We always need to load Tex.pool binding and possibly LaTex.pool as well in the beginning. LaTeXML packages are just like the style and class files in TeX and they have an extension of *.ltxml*. We load our LaTeXbinding juts like how we load our class file in TeX. In our case we would like to load *doc.cls.ltxml*, and therefore we use $\backslash documentclasss\{doc\}$, similarly, if you would like to load a doc type doc.sty.ltxml, you only need to include $\backslash use\ package\{doc\}$ in your .tex file.

2.3 Construction & Constructors

As we are interested in the conversion to XML, we need to understand how constructors works.

DefConstructor *($prototype, $replacement, $options)*

DefConstructor(' \section', "<mock:section><mock:title>#1</mock:title>");
The prototype is the control sequence you have defined in your *.tex* file, and the replacement is what you want it to be built in your XML file. There are options that need to be passed. In our doc.cls.ltxml, it is important to include

beforeDigest =<sub { AssignValue(inPreamble =<0);}

for it makes sure that there spaces between words in the generated XML file. For other similar options, you might not necessarily be able to find them in the manual, albeit, you can go one step further by looking at the other built in bindings such as *article.cls.ltxml* and the *pool* package.

One other error you might encounter when creating a biding from scratch is the constructor for new line. The macro for new line in TEXis \\.
It does not necessarily mean you are able to customize \\, due to some predefinition in the *pool* package, which explains why we change our the conventional \\ in *mockDoc* into *\newline*. The same method can be considered as one possible solution for some *malformed* errors.

2.4 Document Model

After customizing how TEXis translated into XML. There are three more schema to include: RelaxNGSchema, RegisterNamespace, and Tag.

2.4.1 RelaxNGSchema

The constructors tell LaTeXML to add a replacement into XML when it detects a prototype in the *.tex* file. The question is it doesn't tell where should the tag be closed, and therefore we introduce

Tag($Tag,$properties)

meaning whenever there is new paragraph, the last paragraph tag will be closed, before the new paragraph tag can be added. RelaxNGSchema tells how the whole document class in constructed, such that when if we want to create our own document structure, RelaxNGschema is quite important.
LaTeXML is unable to process the compact form which has an extension of .rnc but .rng. It makes your life easier to write in compact form nevetnhelress. The trick it convert your .rnc in to .rng using trang. In a schema file, pretty much you are defining in what elements can be included in different bodies. For example

paragraph = element paragraph { title, p }
section = element section {title,(p —subsection)}*
title = element title { text }
p = element p { (text—element break { empty })}*

The first line tells LaTeXML, in a paragraph it has two elements that are titles and a p. The second line tells in a section, there is a title followed by two possible
structure either title or subsection.

Finally we come to last part of our LaTeXML, namespace. Namespace differentiates our customized macros with others by adding a prefix to our LaTeXML constructor patterns.

RegisterNamespace($prefix, $url)

Congratulations for being able to follow the tutorial to the end. You should have everything you need for a minimal document LaTeXML binding. After the processing the makefile, you should be able to see the generated XML in your current directory!

Reference Bruce R. Miller, November 21, 2014 *LaTeXML The Manual*