A computer cannot yet generate interesting literature

Writing a text that makes interesting literature is not yet something a computer is able to do. A computer has no imagination, no sense of humor so it would produce very boring book.

On the other hand, computers are very good at

- Reproducing text from some electronic source like a database
- extracting interesting sections from other documents
- formatting that output into a pleasing form.

This can all be done with commercial packages but it is also surprisingly simple with \LaTeX.
Some examples

Some examples of such documents are

- Bills for your customers or members. Very handy for your club.
- Requirements items, including implementation of fulfillment state, extracted from a requirements database (such as trac or bugzilla). Very nice in a (software) project team.
- All kinds of reports from databases (Tabular info). What about a good looking inventory lists.
- “customized” output to client specification. Club cards, table card, what not.
- A combination of the above.

Why bother doing it with \LaTeX?

Doing it with \LaTeX

- is free as in free beer.
- provides absolute freedom. There is no limit to what you can do on paper or on screen.
- can be fully automated. There is no need to start up a program and to have someone click the go button, which makes it ideal for repeated jobs on servers.
- is fun. You go into the dangerous world of getting smarter.

\TeX\ and \LaTeX\ macros

Some history

- \TeX\ is written by Prof. Donald Knuth because he was not pleased with the rendering possibilities of publishers for mathematical oriented books. He devised \TeX\ for that purpose and put it in the public domain.
- \LaTeX\ was built upon the macro facility in \TeX\ by Leslie Lamport. This gained a great popularity since the mid eights of the last century.
- Since then, many macro packages were build upon and extended the work of these guys.
The anatomy and use of a macro

A macro can be defined and redefined and subsequently be used.

\%\% \newcommand{\macro}{\textbf{wrap line}}
\textbf{Hello #1}. You owe me \euro\#3 for the item \textit{#2}. I sold you yesterday. Payment is due on \#4. Please pay in time, because my shakeout team is already quite busy.
\%\% add some graphics
\includegraphics[width=10mm]{../../figures/scott_the_pirate}
\includegraphics[width=20mm]{../../figures/scott_the_pirate}
}

This macros defines 4 parameters, used inside its body. You would call it like this:

\macro{John Doe}{BMW 535 (1500,--)}{September 29\textsuperscript{th}}{2012}

As you can see, you can even use comments inside the macro call (in this case to break up the line).

The anatomy and use of a macro, the result

This renders to:

Hello John Doe, You owe me €1500,— for the item BMW 535 I sold you yesterday. Payment is due on September 29\textsuperscript{th} 2012. Please pay in time, because my shakeout team is already quite busy.

This could become quite handy if you want remind your “friends” of any outstanding depts. Print on those yellow Postit\textsuperscript{TM} and stick them wherever your friend lingers.

Other possibilities

Of course the previous silly example is of little real value, but it illustrates the use of templates. Much like the touted templating frameworks that are so popular in the Web world.

This is the way I produce the tablecards, one macro of 35 lines of latex code and some definitions. It is called like this:

\tablecard{Scott the Pirate}

\tablecard{Candy sucker}

\tablecard{Boss of the cot}
The \LaTeX{} API

There is no official programmatic interface to \LaTeX{}. It is simply text with some structuring rules, very much like HTML.

- This makes it very easy to produce the required output.

However, some object oriented programming can make it a bit easier or rather: less error prone.

- This can be done in any (OO) language. I selected Java as an example. With a scripting language (3 P’s) you can have just as much fun.

Any serious project needs at least one GoF pattern

And indeed: there are two fitting to this job:

- Composite fits the tree like structure of documents, much the same as in HTML.
- The template method we use and override is \texttt{toString()} with the primitive operations \texttt{getHead().toString()} and \texttt{getTail().toString()}
Your imagination is limitless

Of course the previous \LaTeX{} example produced by the Main class is not very useful.

But this simple package can already create books and reports with chapters, sections etc, as well as floating stuff like figures and tables.

And think of the possibilities:

- Use the macro facilities: a canned template in combination with repetition.
- Use a \texttt{jdbc} connection and a few extra classes like a \texttt{LongTable} and a \texttt{SummingTable}.  

1\text{Left as an exercises}
My imagination is limitless

Peerweb\textsuperscript{2} does a few of these things like a report on your presence to this colloquium series, the presence list you are currently filling in and of course the famous peerweb tablecard.

Peerweb tablecard

\textsuperscript{2}Peerweb uses PHP and Perl with the same approach to the same effect

Template engines all arround

Text templating engines, that is engines that substitute specially marked text with values are available to many applications and programming languages too.

Think of it als mail-merge on steroids. The programmatic power of the programming language combined with the flexibility of \LaTeX is great.

What follows is a little example in \texttt{freemarker}, an open source (hence free) templating engine that can be used for any text format (think HTML) and indeed also with \LaTeX.

The main ingredients

To use a template you need three things:

1. A template. This is the text you want to fill in at the places so marked.
2. The data source that is used for the filling. Think strongly about data base.
3. The library to merge the two.
4. And some glue code to mix it all up nicely in your application.
Basic freemarker example

The example below is self explaining and simple enough.

```latex
\chapter[Welcome $\{\text{user}\}$ if user == "Big Joe
to our beloved leader.]{Welcome $\{\text{user}\}$ if user == "Big Joe", our beloved leader.}

Our latest product:
```

```latex
\item \{\href{\{\text{latestProduct.url}\}}{\{\text{latestProduct.name}\}}\}
\end{itemize}
\end{document}
```

Yes, you have for and loop constructs as well. The library in this example is freemarker.

Trivial example glue code

```java
/** Get or create a template */
Template temp = cfg.getTemplate("test.ftl");

/** Create a data-model */
Map root = new HashMap();
root.put("user", "Big Joe");
Map latest = new HashMap();
latest.put("latestProduct", latest);
latest.put("url", "products/greenmouse.html");
latest.put("name", "green mouse");

/** Merge data-model with template */
Writer out = new OutputStreamWriter(System.out); 
out.flush();
```

Wrap up

\LaTeX{} has a thriving community, especially among mathematicians and to a lesser extend software engineers. \LaTeX{} is able to produce pleasing results from machine generated data without any further intervention of a human. There is no need to start a GUI, thus making it ideal for server generated documents.

With simplicity of use comes power. Especially combining it with programming.

If you are smart enough, you can do a lot for free. However: beware of the endless refinement syndrome. And never try to automate a "single run" application. Use a (free) office application for that.
Unmentioned possibilities

You might currently very well be using file formats that contain interesting pieces of documentation. For instance:

- The Rational Rose format (the .mdl file) of the version we use at our institute is a text base format (no not XML) that is parse-able. If you put documentation (e.g. use case descriptions) you can extract that with a tool (Perl, Python or something else that handles regular expressions and write that into LaTeX form.

- Something similar applies to the OMG endorsed XMI format. This is the newer UML model exchange standard.

Unmentioned possibilities II

Another possibility is used intensively in our institute:

- Maintaining one set source of LaTeX Beamer slides in 3 languages. This is very helpful in keeping all our multilingual course material in sync. Ask Dr Van Odenhoven. He is an expert at it. It is by the way the method we used to translate the Vigenschow book on Testing.

And of course, there is much more possible than what we can mention here. And all for the software license price of a free beer. (Which is, by the way, what we charge for advice on LaTeX.)

Cheers.

Questions?