LaTeX-on-HTTP: commoditize LaTeX as a cloud service for application developers

Yoan Tournade

Abstract

Although LaTeX is widely used in academia and education, only a few developers use it to edit PDFs in web applications or IT systems. This is strange when we think that lot of developers are well exposed to these academic and scientific milieux — and that many recognize LaTeX for its superior typesetting qualities.

In this paper, we try to answer two questions:

- Why LaTeX use in IT systems is not widespread — and not even common?
- How can we change this state of affairs?

We give an overview of LaTeX-on-HTTP, a tentative to commidize LaTeX use in web applications or IT systems, and to ease its adoption by modern developers.

1 Introduction

LaTeX has its indisputable niches, like acamedic and scientific publications. It is more and more used and accessible for the layman computer user to complete various edition chores, thanks to web projects like *Overleaf*. It has also become a reference for résumé creation for technical professionals.

Isn't that strange then that one of the population the most exposed to LaTeX and the most likely to credit its greatness, *the computer engineers*, do *not* use — and maybe do not even think to use — LaTeX in the applications they develop.

Currently a modern application developer that need PDF edition for a project will found many solutions by searching the web:

1. HTML/CSS to PDF converters;
2. instruction-based PDF generators;
3. headless calls to browser or office softwares.

Browsing a couple of *Stack Overflow* entries will give our developer a ready to be copy-pasted first working code sample, in its language of interest. LaTeX will no appear in the results; and will most certainly not pop up in our developer mind as an eligible tool.

While all these solutions provide decent results, we argue they are far off the typesetting quality attainable by LaTeX.

We can legitimate the prevalence of the first class of solutions by the opportunity to leverage existing HTML and CSS code, but this can't explain why LaTeX would be virtually absent from the common set of solutions.

2 Why LaTeX hasn't permeated modern applications for PDF edition?

We explain this state of affairs from three facets: techniques, customs and knowledges.

2.1 Technical barriers: the not so accessible LaTeX runtime

It is rather easy to install a LaTeX distribution on a personal computer. Modern services like *CoCalc* or *Overleaf* have even removed this need and drastricaly reduced the time required to first interact with LaTeX, providing convenience. Tools like *MathJax* or *LaTeX Base* have even demonstrated that LaTeX can be run in a browser.

However the requirements of an application developer are different: he wants indeed convenience, but more importantly he needs reproducibility and scaling — and he certainly does *not* want to battle with its infrastructure team to explain why it would be pertinent to add many hundred of megabytes of complex runtime requirements to generate PDFs in their application servers.

When the developer has only several hours, or at best days, to automate PDF edition, it will shy away by the complexities of adding the LaTeX runtime in his application, and go for the safer and more recognized solutions.

2.2 Cultural glass walls: mental buckets of typical uses

Many developers, coming or having been exposed to acamedic and scientific milieux, know of LaTeX and recognize its typesetting superiority. They are the people who could shame one of their mates for *not* using LaTeX in their latest technical publication.

Nevertheless for them LaTeX is in a mental bucket that does not permeate with their web application developments or IT system integrations activities.

More than not, they will not even think about LaTeX for editing a PDF document in their application: a glass wall of customs is erected between their need and the LaTeX solution.

2.3 Imperfect knowledge bases: learning by copying

Even if our developer thought about LaTeX as a prospective solution, how could it learn about its implementation?

As a developer, we do not try to reinvent the wheel at each task given to us. We search and we found shared knowledge about recommended tool usage and common patterns. Basically, we use the

more readily available and copy-pasteable code samples for our need; and tweak our custom solution from there.

Of course our developer could easily find a proper LaTeX template to start from for his PDF application — there are great resources on the web for that. But then? Our developer will need the code to compile their LaTeX template to a PDF — and that, they will not find on the web. Compile a LaTeX document on my computer or on the web? Easy. Compile a LaTeX template from a web application — beeing from a server or from any browser — in a couple of line of codes and without adding potentially problematic requirements? Not so easy.

Few pertinents examples will lead our developer to a way to implement LaTeX in his application. If he has the time to be curious and is up for a challenge, he will for sure find a path. This is especially so in the niches where the needs for very high-quality or specialized document edition will justify the cost of bringing LaTeX in the infrastructure. But this is not the typical case which interest us: the common modern web developer that at first just wants a rather simple PDF document in his application.

3 Introducing LaTeX-on-HTTP

The need to create LaTeX-on-HTTP has first emerged when I first converted my quotes-and-invoices consulting templates to LaTeX. I was really glad of the result, which included some basic automation, but I wanted the other members of the team to be able to edit these PDF documents — without having to install several hundred of megabytes of additional software, or to require support to explain why the system complained about a missing CTAN package (which is all very foreign for anyone not introduced to LaTeX, and should not be required to compile a fixed template).

Thus the basic requirements for LaTeX-on-HTTP were rather simple: take the TeX Live runtimes, put them on a server, add an HTTP-based API between the user and the server so we can pass the files to be compiled. *Voilà* — now my team just need an internet access to generate their PDF documents, and they may not even know that they use LaTeX.

3.1 HTTP: the web *lingua franca*

The HTTP API is the important part here; it gives us several researched properties:

- as HTTP is ubiquitous in modern applications, the service is accessible from most of the technical stacks (no more need for complex runtime requirements, a simple HTTP client suffice);
- the verb of HTTP is very familiar to developers;
- it advertises the solution as intended for automation and integrations.

3.2 Hello world: normalizing a compilation job

Let's now present some of the intricaties of the actual implementation by looking at a simple example.

The following JSON code is sent in a POST HTTP request to the `/builds/sync` endpoint, which will return in response a PDF file (when the compilation succeed):

```
{
"compiler": "lualatex",
"resources": [
    {
        "main": true,
        "content": "\\documentclass{article}
        \n \\usepackage{graphicx}\n
        \\begin{document}\n Hello World\\\\ \n
        \\includegraphics[height=2cm]{logo.png}
        \n \\end{document}"
    },
    {
        "path": "logo.png",
        "url": "https://www.ytotech.com/static/
        images/ytotech_logo.png"
    }
]
}
```

You can try a similar example on the command line of your computer by using `curl`; a snippet is available on the project page: `github.com/YtoTech/latex-on-http`.

As you can observe, we pass two main things to the LaTeX-on-HTTP server:

- a set of resources — or source files — to be compiled;
- the engine selected — and other potential options — to control the compilation environment.

You can also remark that we can provite different means to transfer resources to be compiled: by passing the string content directly, by providing an URL pointing to a file — we can imagine and provite several other ways, for convenience and different use cases.

This tentative to normalize a LaTeX compilation job input is important for ensuring the reproducibility of the process.

3.3 Real-life example: editing a letter

A developer can take this API to construct a real-life exemple. Let's say our developer wants to edit a personal letter from the bank IT system, notifying customer of their account opening. Assuming Python is used, the process could look like the following code:

```
# Open and read the template LaTeX file.
with open("opening-letter-template.tex") as f:
    t_str = f.read()

# Replace the dynamic content.
t_str = t_str.replace(
    "$letter-title$", "Your account...")
t_str = t_str.replace(
    "$letter-body$", "Dear Mr...")

# Generate the PDF file,
# using an HTTP client (requests).
r= requests.post(
    "https://latex.ytotech.com/builds/sync",
    json={
        "compiler": "pdflatex",
        "resources: [
            {
                "main": true,
                "content": t_str
            }
        ]
    }
)

# Do something with output PDF file.
with open("opening-letter.pdf", "wb") as f:
    f.write(r.content)
```

The templating could have been done with LaTeX tools, but it is just easier and faster for our developer to interpolate the dynamic content in his native language.

From that simple base example, our developer can easily inject his application data in the letter to be edited; and he can takes the resulting generated PDF to return it in a web page, in his own application API or store it in a storage system.

4 Conclusion

By reducing the runtime requirement to use LaTeX to a familiar HTTP API, we remove maybe the main friction preventing developers to adopt LaTeX in their applications for PDF document edition.

To bolster the prospective success of the solution, we must now engage in a process to document and to publicize this usage of LaTeX in web applications and IT systems — and hope we'll see better typesetted documents in our daily applications in the future as a result.

◇ Yoan Tournade
Soubeyrac
Le Laussou, 47150
France
y (at) yoantournade dot com
https://yoantournade.com