XML to PDF (via CLD)

Aditya Mahajan

15th ConTeXt meeting
24th Sep 2021

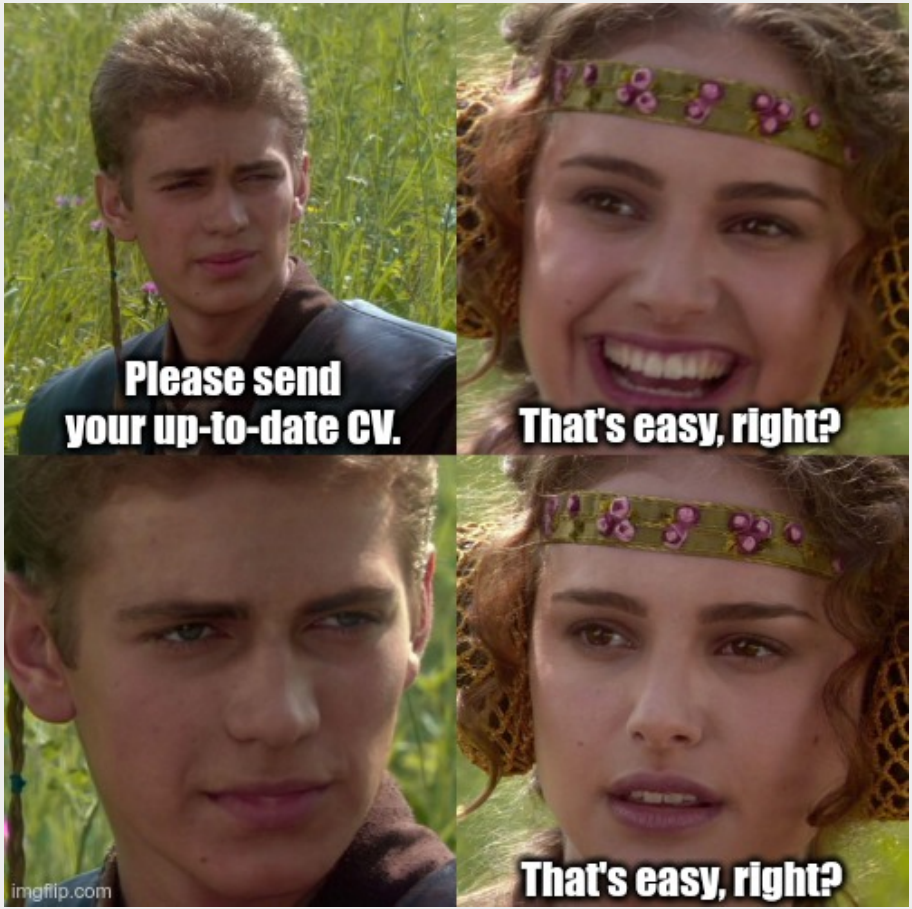
Wait! Why do you want to do that?

Wait! Why do you want to do that?

- It did not start out this way. So, let me provide some background....

Wait! Why do you want to do that?

- It did not start out this way. So, let me provide some background....
- Started in academia 11 years ago.
- Quickly realized that it is extremely important to have an up-to-date CV.
  - Funding agencies
  - award committees
  - yearly evaluations . . .



XML => CLD => PDF–(Aditya)

Different types of CV

Full Academic CV (20 to 50 pages)

- Details of everything (and I mean everything) that you have done professionally
- Presented in visually easy to parse format (tables, lots of tables)

Short CV (5 to 10 pages)

- Important details and summary of professional activities.
- Presented in a compact manner (bullet lists, lots of bullet lists)

Different types of CV

Full Academic CV (20 to 50 pages)

- Details of everything (and I mean everything) that you have done professionally
- Presented in visually easy to parse format (tables, lots of tables)

Short CV (5 to 10 pages)

- Important details and summary of professional activities.
- Presented in a compact manner (bullet lists, lots of bullet lists)

- Everyone maintains multiple Word files, which doubles the work
- I wanted to use a single TeX file. It's easy with careful use of modes.

Different types of CV

Full Academic CV (20 to 50 pages)

- Details of everything (and I mean everything) that you have done professionally
- Presented in visually easy to parse format (tables, lots of tables)

Short CV (5 to 10 pages)

- Important details and summary of professional activities.
- Presented in a compact manner (bullet lists, lots of bullet lists)

- Everyone maintains multiple Word files, which doubles the work
- I wanted to use a single TeX file. It's easy with careful use of modes.
- Key-word driven user interface ...

```
\Grant[ title={...},
        PIs={...},
        year={2011-2015},
        amount={year1, year2, ...},  ]
```

XML => CLD => PDF–(Aditya)

Example output

Example output

mode=full

Project Information	Amount	Personal	Period
Fancy title of the grant, **Funding agency**, Applicant 1 (PI), Applicant 2, and Applicant 3.	50,000	25,000	2018
	50,000	25,000	2019
	50,000	25,000	2020
Fancy title of 2nd grant, **Funding agency**, Applicant 1 (PI), Applicant 2, and Applicant 3.	150,000	20,000	2020
	150,000	20,000	2021
	150,000	20,000	2022
...	...	...	...
	...	...	...
	...	...	...
Total	**600,000**	**135,000**	

Example output

mode=short

- *Fancy title of the grant*, **Funding agency**, Applicant 1 (PI), Applicant 2, and Applicant 3. **$150,000** (2018–2020).
- *Fancy title of 2nd grant*, **Funding agency**, Applicant 1 (PI), Applicant 2, and Applicant 3. **$450,000** (2018–2020).
- . . .
- . . .
- . . .

Different types of CV (continued)

Funding agencies

- Similar to short CV, but only list grants and pubs from the last 6 years.

Different types of CV (continued)

Funding agencies

- Similar to short CV, but only list grants and pubs from the last 6 years.

Another funding agency . . .

- . . . 5 years . . .

Different types of CV (continued)

Funding agencies

- Similar to short CV, but only list grants and pubs from the last 6 years.

Another funding agency . . .

- . . . 5 years . . .

- I can still handle it with conditional processing . . .

Different types of CV (continued)

Funding agencies

- Similar to short CV, but only list grants and pubs from the last 6 years.

Another funding agency . . .

- . . . 5 years . . .

- I can still handle it with conditional processing . . .

```
\ifnum\currentpaperparameter\c!year
          > \numexpr\currentyear - \interval\relax
  ....
\fi
```

- . . . with key-value interface

```
\publication[  title={....},
               authors={author1, author2, ...},
               journal={...}, ... ]
```

Different types of CV (still continued . . .)

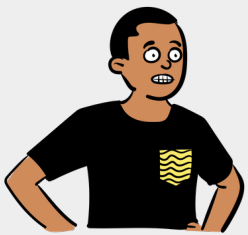
Yearly review

- It will be nice if you include tables summarizing the grant record, publication record, and supervision record.

Different types of CV (still continued . . .)

Yearly review

- It will be nice if you include tables summarizing the grant record, publication record, and supervision record.



- This is just arithmetic and TeX is Turing complete, right?
- Writing clean code in TeX is hard. So, moved all calculations to Lua.

Different types of CV (still continued ...)

Yearly review

- It will be nice if you include tables summarizing the grant record, publication record, and supervision record.

- This is just arithmetic and TeX is Turing complete, right?
- Writing clean code in TeX is hard. So, moved all calculations to Lua.
- But why not move the interface to Lua as well ...

```
local publications = {
    { ["title"] = {....},
      ["authors"] = { "author1", "author2", "author3" },
      ["journal"] = {...},
    },
    { ... },
}
```

Different types of CV (there is more!!)

Grant review

- In the list of publications, add an asterix next to the student working under your supervision...

Different types of CV (there is more!!)

Grant review

- In the list of publications, add an asterix next to the student working under your supervision...

- This is now more of an interface design question.

```
local publications = {
    { ["title"] = {....},
      ["authors"] = {
          { ["name"] = "author1", ["status"] = "supervised" },
          { ["name"] = "...",     ["status"] = ",,," },
      },
    },
    { ... },
}
```

Finally, had a system that met all the requirements.

Finally, had a system that met all the requirements.

- But data entry was error prone . . .
- Forgot quotes around keys or values, forget to add a key, etc.

Finally, had a system that met all the requirements.

- But data entry was error prone ...
- Forgot quotes around keys or values, forget to add a key, etc.
- Started thinking about writing a `validate` function in Lua ...
  ... but realized that this is a solved problem in XML.
- Write a schema in RNG and verify using `jing` (or other tools).

Adding XML to the mix

- ConTeXt XML mode was being rewamped around this time.
- Filtering based on LPATH or CSS selectors ...

  ... but I could't wrap by head around it.

Adding XML to the mix

- ConTeXt XML mode was being rewamped around this time.
- Filtering based on LPATH or CSS selectors ...
  ... but I could't wrap by head around it.

- I already had a working system for going from LUA to PDF ... so all I needed to do was convert XML-tree to LUA table.
- `lmxl.loadfile(...)` family of functions already do that ...
  ... I just need to do some data munging.

Time for some reverse engineering

Simple experiment

Simple experiment

```
local example = [==[
<?xml version="1.0" encoding="UTF-8" ?>
<publication-list>
    <journal-list>
      <publication>
        Paper 1
      </publication>
    </journal-list>
</publication-list>
]==]

local xml_data = lxml.loaddata("publications", example)
```

Simple experiment

```
t={
 ["at"]={},
 ["dt"]={
  {
   ["dt"]={ "xml
version=\"1.0\" encoding=\"UTF-8\" ",
" },
   ["ns"]="",
   ["special"]=true,
   ["tg"]="@pi@",
  },
  "\
",
  {
   ["at"]={},
   ["dt"]={
    "\
",
  {
   ["at"]={},
   ["dt"]={
    "\
",
   {
    ["at"]={},
    ["dt"]={ "\
",
     Paper 1\
" },
    ["ni"]=2,
    ["ns"]="",
    ["rn"]="",
    ["tg"]="publication",
   },
  "\
",
   },
   ["ni"]=2,
   ["ns"]="",
   ["rn"]="",
   ["tg"]="journal-list",
  },
  "\
",
  },
  ["ni"]=3,
  ["ns"]="",
  ["rn"]="",
  ["tg"]="publication-list",
 },
},
```

Filtering . . .

Filtering . . .

```
local example = [==[
<?xml version="1.0" encoding="UTF-8" ?>
<publication-list>
    <journal-list>
        <publication>
            <title>Paper 1</title>
        </publication>
    </journal-list>
    <conference-list>
      <!-- -->
    </conference-list>
</publication-list>
]==]

local xml_data = lxml.loaddata("publications", example)
local journal_data = lxml.filter(xml_data, "journal-list/publication")
```

Filtering . . .

```
t={
 {
  ["at"]={},
  ["dt"]={
   "\
            ",
   {
    ["at"]={},
    ["dt"]={ "Paper 1" },
    ["ei"]=1,
    ["en"]=0,
    ["ni"]=2,
    ["ns"]="",
    ["rn"]="",
    ["tg"]="title",
   },
   "\
        ",
  },
  ["ei"]=1,
  ["en"]=1,
  ["mi"]=1,
  ["ni"]=2,
  ["ns"]="",
  ["rn"]="",
  ["tg"]="publication",
 },
}
```

Attributes . . .

Attributes . . .

```
local example = [==[
<?xml version="1.0" encoding="UTF-8" ?>
<publication-list>
    <journal-list>
        <publication status="submitted">
            <title>Paper 1</title>
        </publication>
        <publication status="appeared">
            <title>Paper 2</title>
        </publication>
    </journal-list>
</publication-list>
]==]

local xml_data = lxml.loaddata("publications", example)
local journal_data = lxml.filter(xml_data, "journal-list/publication")
```

Attributes . . .

```
t={
 {
  ["at"]={
   ["status"]="submitted",
  },
  ["dt"]={
   "\
            ",
   {
    ["at"]={},
    ["dt"]={ "Paper 1" },
    ["ei"]=1,
    ["en"]=0,
    ["ni"]=2,
    ["ns"]="",
    ["rn"]="",
    ["tg"]="title",
   },
   "\
        ",
  },
  ["ei"]=1,
  ["en"]=1,
  ["mi"]=1,
  ["ni"]=2,
  ["ns"]="",
  ["rn"]="",
  ["tg"]="publication",
 },
 {
  ["at"]={
   ["status"]="appeared",
  },
  ["dt"]={
   "\
            ",
   {
    ["at"]={},
    ["dt"]={ "Paper 2" },
    ["ei"]=1,
    ["en"]=0,
    ["ni"]=2,
    ["ns"]="",
    ["rn"]="",
    ["tg"]="title",
   },
   "\
       ",
```

The main idea

Load the XML file

```
local xml_pubs_list = lxml.load("publications", "publications.xml")
```

The main idea

Load the XML file

```
local xml_pubs_list = lxml.load("publications", "publications.xml")
```

Filter the appropriate list

```
local xml_journals = lxml.filter(xml_pubs_list, "journal-list/publication")
```

The main idea

Load the XML file

```
local xml_pubs_list = lxml.load("publications", "publications.xml")
```

Filter the appropriate list

```
local xml_journals = lxml.filter(xml_pubs_list, "journal-list/publication")
```

Munge data

```
local lua_journals = munge_publications(xml_journals)
```

The main idea

Load the XML file

```
local xml_pubs_list = lxml.load("publications", "publications.xml")
```

Filter the appropriate list

```
local xml_journals = lxml.filter(xml_pubs_list, "journal-list/publication")
```

Munge data

```
local lua_journals = munge_publications(xml_journals)
```

Process data using old CLD code

```
typeset_journals(lua_journals)
```

Example of data munging

```
local munge_publications = function (xml_pubs)
  local lua_pubs = {}

  for i = 1, #xml_pubs do
    local xml_current_pub = xml_pubs[i]
    local lua_current_pub = { }

    lua_current_pub.status    = xml_current_pub.at.status
    lua_current_pub.title     = cv.extract(xml_current_pub, "title")
    lua_current_pub.authors   = pubs_extract_authors(xml_current_pub, "authors")
    lua_current_pub.journal   = cv.extract(xml_current_pub, "journal")
    ...
    lua_pubs[i] = lua_current_pub
  end

  return lua_pubs
end
```

Example of data munging

```
local munge_publications = function (xml_pubs)
  local lua_pubs = {}

  for i = 1,
    local xml
    local lua

    lua_curre
    lua_curre
    lua_curre                                          "authors")
    lua_curre
    ...
    lua_pubs[i] = lua_current_pub
  end

  return lua_pubs
end
```

```
function cv.extract(data, field)
    local extracted_field = lxml.filter(data, field)

    if extracted_field ~= nil then
        return string.trim(extracted_field[1].dt[1])
    end
end
```

So, what does the output look like

Conclusion

Further automation

- Use ConTeXt to generate list-of-publication etc on my webpage.
- Use `inotify`-based watchers to automatically compile different versions of the CV and my webpage whenever an XML file is changed.
- Have been running this setup for almost 10 years. Works flawlessless.

Conclusion

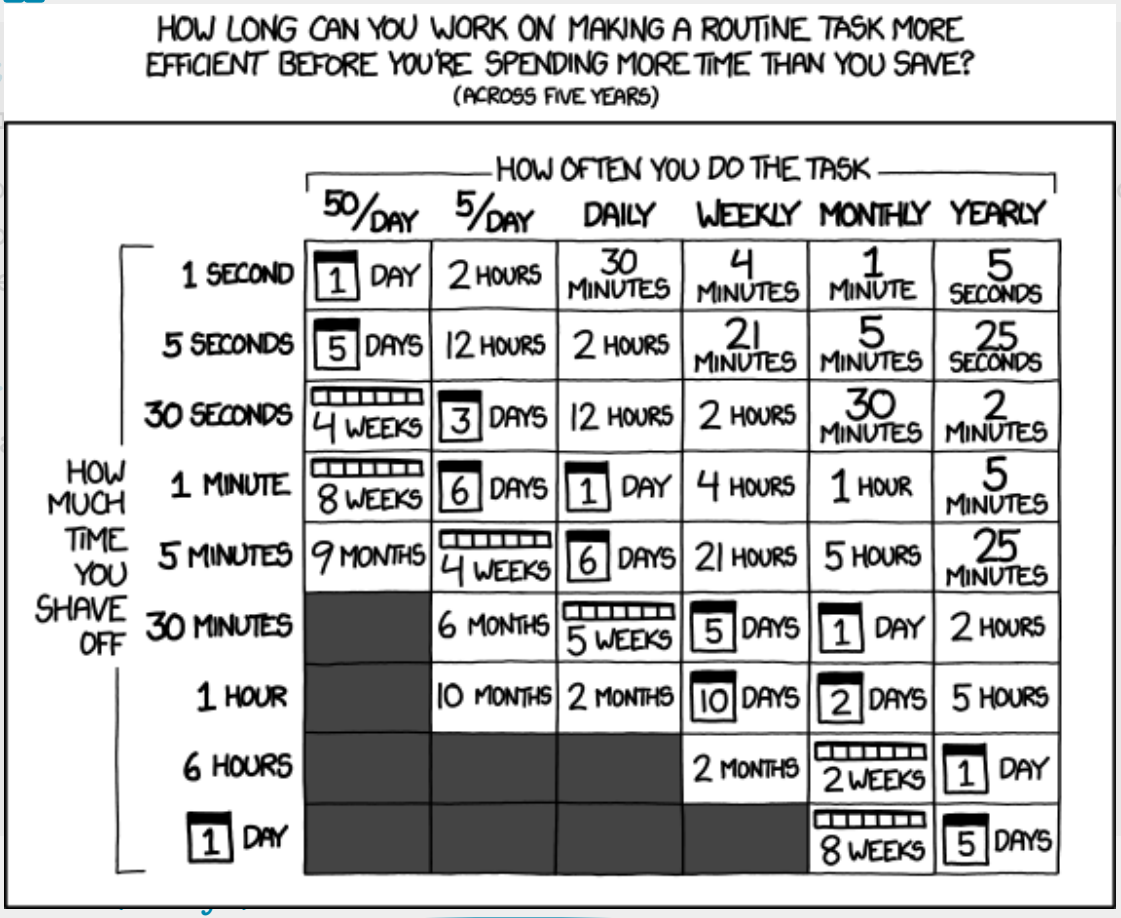
Further automation

- Use ConTeXt to generate list-of-publication etc on my webpage.
- Use `inotify`-based watchers to automatically compile different versions of the CV and my webpage whenever an XML file is changed.
- Have been running this setup for almost 10 years. Works flawlessless.

Complexity

- Less than 3000 lines of clean Lua code.

Conclusion


HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)
HOW OFTEN YOU DO THE TASK
50/DAY
5/DAY
DAILY
WEEKLY
MONTHLY
YEARLY
HOW MUCH TIME YOU SHAVE OFF
1 SECOND
1 DAY
2 HOURS
30 MINUTES
4 MINUTES
1 MINUTE
5 SECONDS
5 SECONDS
5 DAYS
12 HOURS
2 HOURS
21 MINUTES
5 MINUTES
25 SECONDS
30 SECONDS
4 WEEKS
3 DAYS
12 HOURS
2 HOURS
30 MINUTES
2 MINUTES
1 MINUTE
8 WEEKS
6 DAYS
1 DAY
4 HOURS
1 HOUR
5 MINUTES
5 MINUTES
9 MONTHS
4 WEEKS
6 DAYS
21 HOURS
5 HOURS
25 MINUTES
30 MINUTES
6 MONTHS
5 WEEKS
5 DAYS
1 DAY
2 HOURS
1 HOUR
10 MONTHS
2 MONTHS
10 DAYS
2 DAYS
5 HOURS
6 HOURS
2 MONTHS
2 WEEKS
1 DAY
1 DAY
8 WEEKS
5 DAYS

Conclusion

Further automation

- Use ConTeXt to generate list-of-publication etc on my webpage.
- Use `inotify`-based watchers to automatically compile different versions of the CV and my webpage whenever an XML file is changed.
- Have been running this setup for almost 10 years. Works flawlessless.

Complexity

- Less than 3000 lines of clean Lua code.

But . . .

- The funding agencies decided to come up an online system for creating CV.
- You enter all the information in a web-based system and it generates an appropriately formatted CV for you.
- Must include a "common CV" when submitting grants. It is UGLY!!!