CONTEXT

al calendar CONTEXT User Module

Interval Calendar

Willi Egger

April 15, 2023

Interval Calendar

```
1  \startmodule[intervalcalendar]
2  \writestatus{loading}{ConTeXt User Module / Interval Calendar}
```

Introduction

After announcing a presentation for the 16th CONTEXT-Meeting in Dreifelden, Germany, over a new version of the PocketDiary module, Taco Hoekwater asked whether the PocketDiary module might also be able to produce lists with calendar-dates in intervals like ‘weekly’, ‘twoweekly’ or ‘monthly’ for the purpose of taking notes for checks on a regular base. – The lists should be configurable for the length of the list, the number of columns and the possibility to assign an individual column-header. – Quickly it appeared, that the use of the PocketDiary module is not necessary. In order to be able to do math on dates it is the easiest to convert a date to a timestamp in seconds or what is used by the operating-system. – Due to the fact that there are repeating calculations done it is easiest to keep as much as possible at the lua end. Therefore the out-put is prepared as a CONTEXT-LUA-document (cld).

Interval Calendar

One Mode for Each List

The module can be used for the production of different lists. For each list we define a mode.

```
\definemode[weekly][keep]
\definemode[twoweekly][keep]
\definemode[monthly][keep]
```

None of the modes are activated.

Interval Calendar

General Setups

```
6  \setupbodyfont[dejavu,ss,12pt]
7  \setuppagenumbering[location=]

8  \setuppapersize[A4,portrait][A4,portrait]

9  \setuplayout
10   [topspace=20mm,
11   backspace=12mm,
12   header=10mm,
13   footer=2\bodyfontsize,
14   height=middle,
15   width=middle]

16 \setupfootertexts
17   [\jobname .pdf]
18   [\pagenumber\ / \totalnumberofpages]
```

LUA-Code

```
\startluacode

  thirddata          = thirddata     or { }
  thirddata.calendar = { }
  local calendar     = thirddata.calendar

  local report = logs.reporter("Interval Checks")

  local interval_dates = {}

  function calendar.interval_checks(startdate,stopdate,cols,intval)

    local startdate_string =startdate
    local stopdate_string  =stopdate
    local columns=cols
    local intervaldays = intval

    local year,month,day = calendar.date(startdate_string)
    local start_time_stamp = os.time({year=year,month=month,day=day})

    --report("Starttimestamp: %s", start_time_stamp)

    local year,month,day = calendar.date(stopdate_string)
    local end_time_stamp = os.time{year=year,month=month,day=day}

    local interval = intervaldays * 24 * 60 * 60

    for i = start_time_stamp, end_time_stamp, interval do
      local interval_date = os.date("%d-%m-%Y",i)
      table.insert(interval_dates,interval_date)
      report("%s",interval_date)
    end

    calendar.print_interval_checks(columns)
  end

  function calendar.print_interval_checks(cols)

    local columns = cols

    context.bTABLE({setups="table:interval_check"})
      context.bTABLEhead()
        context.bTR()
          for i = 1,columns do
            context.bTH()
              context.labeltext("c"..i)
            context.eTH()
          end
        context.eTR()
      context.eTABLEhead()
      context.bTABLEbody()
      for k,v in ipairs(interval_dates) do
```

```
57          context.bTR()
58            context.bTD()
59              context(v)
60            context.eTD()
61            for i=1,columns-1 do
62              context.bTD()
63                context.strut()
64              context.eTD()
65            end
66          context.eTR()
67        end
68      context.eTABLEbody()
69      context.eTABLE()
70    end

71    function calendar.date(inputstr)

72      --report("Input : %s",inputstr)

73      local sep = "%-%s/"
74      if sep == nil then
75        sep = "%s"
76      end
77      local t={}
78      i=1
79      for str in string.gmatch(inputstr, "([^"..sep.."]+)") do
80        t[i] = str
81        --report("Actual string %s", str)
82        i = i + 1
83      end

84      --report("Datum strings: %s, %s, %s", t[1],t[2],t[3])

85      return t[1],t[2],t[3]
86    end
87  \stopluacode
```

The main function is the ‘calendar.interval_ckecks’ function with 4 parameters. The function needs a start-date, an end-date, the number of columns for the table in the out-put and the interval in days. The function calculates all dates fitting into the start and end dates based on the epoch-time. It converts the timestamp into a human readable date format and puts the values into a table ‘interval_dates’.

The second function is called from within the first function and gets the number of columns as parameter. This function creates the out-put table. It is a flexible setup, which is based on the number of the columns provided as a parameter. Label-texts are used for the column heads and the column heads are repeated at the top of each following page.

The third function is there to deal with the preparation of the dates, which are sent to LUA as strings. It extracts the year (yyyy) the month (m) and day (d). The extracted values are put into a table. The dates can be entered in the format 2022-8-21 or as 2022/8/21.

Macro for Calling the List Generator

The command needs 4 parameters: Start-date, stop-date, number of columns in the table and the interval in days.

```
88  \define[4]\Checklist{\ctxlua{thirddata.calendar.interval_checks(#1,#2,#3,#4)}}
```

Setups for the Out-Put Table

The layout of the out-put table is organized at the TEX end. If the table design changes in terms of more or less columns, this setup has to be adapted.

```
\startsetups table:interval_check
  \setupTABLE[split=repeat]
  \setupTABLE[r][each][height=12mm,align=lohi]
  \setupTABLE[r][1][style=bold,align={lohi,middle}]
  \setupTABLE[c][1][width=0.2\textwidth]
  \setupTABLE[c][2,3,4][width=0.15\textwidth]
  \setupTABLE[c][5][width=0.3\textwidth]
\stopsetups
```

Column Head Texts

It has been chosen to use the label-text mechanism to implement the head texts of the columns. Depending on the design of the out-put table the names can be adapted, the list can be made longer or shorter according to the needs.

```
\setuplabeltext[en][c1=Datum]
\setuplabeltext[en][c2=Gas]
\setuplabeltext[en][c3=Electricity]
\setuplabeltext[en][c4=Water]
\setuplabeltext[en][c5=Observation]
```

Header Texts of Out-Put

Also for the header texts the label text mechanism is used. Now only three intervals are defined, but this mechanism allows easily to extend the list of possible header texts. It also allows to implement other languages. Depending on the setting of the `\mainlanguage[]` the correct labels are picked up and typeset.

```
\setuplabeltext[en][weekly={Weekly Checks}]
\setuplabeltext[en][twoweekly={Two-weekly Checks}]
\setuplabeltext[en][monthly={Monthly Checks}]

\setuplabeltext[de][weekly={Wochen Checks}]
\setuplabeltext[de][twoweekly={Zwei Wochen Checks}]
\setuplabeltext[de][monthly={Monatliche Checks}]

\setuplabeltext[nl][weekly={Weekelijkse Checks}]
\setuplabeltext[nl][twoweekly={Twee-weekelijkse Checks}]
\setuplabeltext[nl][monthly={Maandelijkse Checks}]
```

Header Text Setup

Each list gets a header text fitting the purpose of the list.

```
\startmode[weekly]
  \setupheadertexts
    [\midaligned{\bfc \labeltext{weekly}}]
    []
\stopmode

\startmode[twoweekly]
  \setupheadertexts
    [\midaligned{\bfc \labeltext{twoweekly}}]
    []
\stopmode

\startmode[monthly]
  \setupheadertexts
    [\midaligned{\bfc \labeltext{monthly}}]
    []
\stopmode

\stopmodule
```