Inhaltsverzeichnis

Gruppierte Exceltabelle mit Python, pyopenxl und xml

Die Aufgabe bestand darin, die in xml vorliegenden ineinander verschachtelten Tasks einer GANT-Projektplanung (http://ganttproject.biz/about) einzulesen, etwas auszumisten und dann in Excel zu übertragen, und zwar so, das möglichst die Hierarchie der Ober- und Unteraufgaben erhalten bleibt.

Am Ende sah das dann letztlich so aus (so als Blaupause, wenn man mal wieder Excel- Gruppierungen braucht) :

```
#!/usr/bin/env python3
import sys
import openpyxl
from datetime import datetime
from xml.etree import ElementTree

def recursive_tasks(parent_task_list, level, allocations):
    result = []
    for task in parent_task_list.findall("task"):
        if task.get("id") not in allocations:
            allocs = []
        else:
            allocs = allocations[task.get("id")]
        result.append(
            {
                "level": level,
                "data": [
                    task.get("name"),
                    datetime. strptime(task.get("start"), '%Y-%m-%d'),
                    int(task.get("duration")),
                    int(task.get("complete")),
                    ", ".join(allocs),
                ],
            }
        )
        result = result + recursive_tasks(task, level + 1, allocations)
    return result


def recursive_ws_group(ws, task_list, position):
    # let's see if we can fix the group problem recursively...
    this_task_level = task_list[position]["level"]
    next_task_level = this_task_level
    while True:
        if position < len(task_list) - 1:
            position += 1
        else:  # end reached
            return position
```

```
        next_task_level = task_list[position]["level"]
        if next_task_level > this_task_level:
            end_level_position = position
            while (
                end_level_position < len(task_list) - 1
                and task_list[end_level_position]["level"] > this_task_level
            ):
                end_level_position += 1
            if (
                end_level_position < len(task_list) - 1
            ):  # we have not reeached the end, so we need to decrement by 1
to point to the correct element
                end_level_position -= 1
            if end_level_position > position:
                ws.row_dimensions.group(
                    position + 2, end_level_position+2,
outline_level=this_task_level + 1 # we need to add one (1) offset more,
because we have an additonal header line in the table
                )
            position = recursive_ws_group(ws, task_list, position)
        if next_task_level < this_task_level:  # end if this level reached
            return position

tree = ElementTree.parse(sys.argv[1])

# ElementTree.dump(tree)
root = tree.getroot()

resources_element = root.find("resources")
resources = {}
for resource in resources_element.findall("resource"):
    resources[resource.get("id")] = resource.get("name")
allocations_element = root.find("allocations")
allocations = {}
for allocation in allocations_element.findall("allocation"):
    task_id = allocation.get("task-id")
    if task_id not in allocations:
        allocations[task_id] = []
    allocations[task_id].append(resources[allocation.get("resource-id")])
tasks = root.find("tasks")
task_list = recursive_tasks(tasks, 0, allocations)
#
https://stackoverflow.com/questions/27133731/folding-multiple-rows-with-open
pyxl
wb = openpyxl.Workbook()
ws = wb.active
# write the header
ws.append(["Task", "Start", "Duration", "Complete", "Resources"])
for task_item in task_list:
    ws.append(task_item["data"])
```

```
ws.sheet_properties.outlinePr.summaryBelow = True
recursive_ws_group(ws, task_list, 0)

wb.save(sys.argv[2])
```