Directory Index Building - 2

engelbert gruber

07.04.2020
0TH GRADE SCENES

INTRODUCTION

We got so far as index by filename and index by directory.

```
#!/usr/bin/python3
"""
Make different index views of directory content
"""

from pathlib import Path
# startverzeichnis "."
p = Path(".")
# alle Eintraege im Verzeichnis und in den Unterverzeichnissen
liste = []
for e in p.iterdir():
    if e.is_dir(): # Unterverzeichnisinhalt holen
        for e1 in e.iterdir():
            liste.append(e1)
        else:
            liste.append(e) # an liste anhaengen

# TODO woher Author, Programmiersprache, etc

# Ausgabe html:
def wr_html_file(filename, liste, get_subtitle):
    prev_e = None
```

```
    with open(filename, "w") as f:
        f.write("<!DOCTYPE html>\n<html>\n")
        f.write("<head>\n")
        f.write("  <meta charset=\"utf-8\">\n")
        f.write("  <link rel=\"stylesheet\" type=\"text/css\"
href=\"style.css\">\n")
        f.write("</head>\n")
        f.write("<body>\n")
        f.write("<ul>\n")
        for e in liste:
            _e = get_subtitle(e)
            if prev_e != _e:
                f.write("</ul><div class=\"sep\">%s</div><ul>\n" % _e)
                prev_e = _e
            f.write("<li><a href=\"%s\">%s</a></li>\n" % (e, e.name) )
            f.write("</ul>\n")
        f.write("</body></html>\n")

# sortiert nach Dateiname
liste.sort(key=lambda p: p.name.lower())
wr_html_file("index-name.html", liste, lambda p: p.name[0].lower())

# Ausgabe html: sortiert nach Datum (das Verzeichnis)
liste.sort(key=lambda p: str(p.parent).lower())
wr_html_file("index-date.html", liste, lambda p: p.parent)
```

Ausgabe html: sortiert ...

Todo: html menu, move system files away, remove empty ul at html start

MENU for all

Move constant text into template

This part (and more) is fix (BUG: we do not have a page title)

```
        f.write("<!DOCTYPE html>\n<html>\n")
        f.write("<head>\n")
        f.write("  <meta charset=\"utf-8\">\n")
        f.write("  <link rel=\"stylesheet\" type=\"text/css\"
href=\"style.css\">\n")
```

```
            f.write("</head>\n")
```

Let's move it into a file ... head-template.html, in a directory ... "_"

```
SYSDIR = "_"
HEAD_TEMPLATE = "head-template.html"

# Ausgabe html:
def wr_html_file(filename, liste, get_subtitle):
    prev_e = None
    with open(filename, "w") as f:
        # TODO path join
        head = open(SYSDIR + "/" + HEAD_TEMPLATE).read()
        f.write(head)
        f.write("<body>\n")
```

The head-template.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
```

Menu into head-template

We put the menu into the template.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <div id="mainmenu">
    <a href="index-name.html">byName</a>
    <a href="index-date.html">byDate</a>
  </div>
```

And remove: `f.write("<body>\n")`

Add a system directory for clean up

Let's move the template and stylesheet into SYSDIR.

We can NOT (easily) move the index files into SYSDIR, because then all paths in the index-html must be relative to the SYSDIR ...

STOP! Clean up the TODO path join. Means: we verbinden wir zwei Pfadelemente ohne dass wir den Trenner "/" vorgeben, wir haben eine riesige pathlib ? Die Doku sagt

> >>> PurePath('/etc', '/usr', 'lib64')

Oder :-) der "/" Operator

> >>> sysdir = PurePath('_')
> >>> sysdir / 'head-template'
> PurePosixPath('_/head-template')

New program start

```
from pathlib import Path, PurePath
# startverzeichnis "."
p = Path(".")
# alle Eintraege im Verzeichnis und in den Unterverzeichnissen
liste = []
for e in p.iterdir():
    if e.is_dir(): # Unterverzeichnisinhalt holen
        for e1 in e.iterdir():
            liste.append(e1)
    else:
        liste.append(e) # an liste anhaengen

# TODO woher Author, Programmiersprache, etc

SYSDIR = PurePath("_")
HEAD_TEMPLATE = "head-template.html"

# Ausgabe html:
def wr_html_file(filename, liste, get_subtitle):
```

```
        prev_e = None
        with open(filename, "w") as f:
            head = open(SYSDIR / HEAD_TEMPLATE).read()
            f.write(head)
            f.write("<ul>\n")
            for e in liste:
```

Und im den stylesheet verschieben, im header und im filesystem

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="_/style.css">
</head>
<body>
  <div id="mainmenu">
     <a href="index-name.html">byName</a>
     <a href="index-date.html">byDate</a>
  </div>
```

Das leere ul entfernen

Beim Schreiben der html-Datei fangen wir immer mit einem Trenner (also fällt das erste ul weg. Das schliessend schreiben wir nur wenn prev_e nicht None ist.

```
prev_e = None
with open(filename, "w") as f:
    head = open(SYSDIR / HEAD_TEMPLATE).read()
    f.write(head)
    f.write("<ul>\n")
    for e in liste:
        _e = get_subtitle(e)
        if prev_e != _e:
            if prev_e != None:
                f.write("</ul>\n")
            f.write("<div class=\"sep\">%s</div><ul>\n" % _e)
            prev_e = _e
```

ADDITIONAL Attributes

Wie dazu speichern

Wie können wir Programmiersprachen und Autoren und ... dazu bauen ?

Python style simplest

```
class ExtraPath:
    author = "system"

p = ExtraPath()
p.author
```

Das soll in unsere `liste` eingebaut werden.

Aber dort haben wir pathlib.Path-Objekte. Das bedeutet wir müssen das als von `Path` abgeleitete Klasse machen.

```
class ExtraPath(pathlib.Path):
    author = "system"

x = ExtraPath()
```

Da kommt aber ein Fehler

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/lib/python3.7/pathlib.py", line 1010, in __new__
      self = cls._from_parts(args, init=False)
  File "/usr/lib/python3.7/pathlib.py", line 665, in _from_parts
      drv, root, parts = self._parse_args(args)
  File "/usr/lib/python3.7/pathlib.py", line 658, in _parse_args
      return cls._flavour.parse_parts(parts)
AttributeError: type object 'ExtraPath' has no attribute '_flavour'
```

__new__ ist eine Funktion die beim Anlegen/Initialisieren eines Objektes aufgerufen wird (denke ich) und google findet pathlib.Path ist nicht erweiterbar.

Wir brauchen nicht so viel von Path das bisschen bauen wir uns dazu.

```
class ExtraPath():
    """
    pathlib.Path (ist nicht erweiterbar) und ein paar Attribute
    """
    def __init__(self, pathlibPath):
        self.path = pathlibPath
        self.name = self.path.name
        self.parent = self.path.parent
        self.author = "system"
        self.complang = self.path.suffix
    def __str__(self):
        return str(self.path)

# alle Eintraege im Verzeichnis und in den Unterverzeichnissen
liste = []
for e in p.iterdir():
        if e.is_dir(): # Unterverzeichnisinhalt holen
            for e1 in e.iterdir():
                liste.append(ExtraPath(e1))
        else:
            liste.append(ExtraPath(e)) # an liste anhaengen
# TODO woher Author, Programmiersprache, etc
```

Und am Ende

```
# Ausgabe html: sortiert nach complang
liste.sort(key=lambda p: str(p.complang).lower())
wr_html_file("index-complang.html", liste, lambda p:
p.complang)
```

Index-complang.html

- serial-read-1

.css

- style.css

.docx

- Datenbanken.docx
- arduino Serial IO.docx
- java notes.docx
- python notes.docx

.html

- index-complang.html
- index-date.html
- Index-name.html

Und ins menu einbauen.

Refactoring

Wir haben die Klasse ExtraPath nich von pathlib.Path abgeleitet, weil das aus irgendwelchen Gründen nicht geht. ExtraPath ist nur ein Wrapper um die Klasse Path.

Bei Vererbung im objektorientierten Kontext, gibt es zwei Hauptbeziehungen.

1. Die neue Klasse ist wie die alte nur etwas anders. Mann und Frau sind beides Personen.
2. Die neue Klasse hat die alte Klasse als Bestandteil. Ein Auto hat Räder ist aber keines.

Sagen wir die “liste” ist nicht eine Liste von Path-Objekten, sondern von Document-Objekten, die einen Path haben.

We rename the class.

```
class Document():
    """
    Document hat ein pathlib.Path (ist nicht erweiterbar) und ein paar
    andere Attribute.
    """
    def __init__(self, pathlibPath):
        self.path = pathlibPath
        self.name = self.path.name
        self.parent = self.path.parent
        self.author = "system"
        self.complang = self.path.suffix
    def __str__(self):
```

```
        return str(self.path)
```

Woher nehmen wir die Attributwerte ?

Aus einer Datei mit dem selben Namen aber der Erweiterung “info” ? die wir in der Funktion Document.__init__ versuchen einzulesen.

Wir schauen ob es eine Datei mit dem selben Namen aber der Erweiterung … “info” gibt, wenn ja lesen wir die Attribute aus.

pathlib.Path hat eine Methode with_suffix die die Dateierweiterung austauscht.

```
>>> import pathlib
>>> p = pathlib.Path("abc.def")
>>> p
PosixPath('abc.def')
>>> p.with_suffix(".txt")
PosixPath('abc.txt')
```

Und exists gibt es auch

```
>>> p.with_suffix(".gh").exists()
False
```

Which format do we use for the info-file ? Gibt es etwas in der stdlib ?

Configparser liest soetwas:

```
[DEFAULT]
ServerAliveInterval = 45
Compression = yes
```

json schaut so aus:

```
{
"author": "system",
 "complang": "python"
}
```

Oder selber … ich mag json probieren.

Dann ändert sich die Klasse Document und leider kommen ein paar TODOs dazu.

```
class Document():
    """
    Document hat ein pathlib.Path (ist nicht erweiterbar) und ein paar
    andere Attribute.
    """
    def __init__(self, pathlibPath):
        self.path = pathlibPath
        self.name = self.path.name
        self.parent = self.path.parent
        self.author = "system"
        self.complang = self.path.suffix
        self.read_attributes()
    def __str__(self):
        return str(self.path)
    def read_attributes(self):
        """
        read attribute file if present.
        """
        iPath = self.path.with_suffix(".info")
        if iPath.exists():
            # TODO with context ?
            a = json.load(iPath.open())
            # TODO how to move all into attributes ?
            if "complang" in a:
                self.complang = a["complang"]
            if "author" in a:
                self.author = a["author"]


    # TODO ignore info files in liste
```

Works ... for now ... add the page

```
# Ausgabe html: sortiert nach autor
liste.sort(key=lambda p: str(p.author).lower())
wr_html_file("index-author.html", liste, lambda p: p.author)
```

Und im Menu

```
  <div id="mainmenu">
```

```
    <a href="index-name.html">byName</a>
    <a href="index-date.html">byDate</a>
    <a href="index-complang.html">byLanguage</a>
    <a href="index-author.html">byAuthor</a>
</div>
```

REFERENCES

1. https://docs.python.org/3/library/pathlib.html
2.