



Université Cheikh Anta Diop



Ecole Supérieure Polytechnique

Département Génie Informatique

Khadi Kama
Rouguiyatou Thiam



Url du dépôt Github

https://github.com/Guiyatou/PROJET_SGBD

Langage choisi: PYTHON

POUR LE FICHER XJ_CONVERTOR

On a importé les modules suivants :

```
import os
import sys
import json
import xml_generation
import json_validation,xml_validation,xml_extraction
```

le **module os** fournit une manière portable d'utiliser les fonctionnalités dépendantes du système d'exploitation .

Le **module sys** permet de communiquer avec le système d'exploitation.

```

if len(sys.argv) == 7:

    fileType = sys.argv[2]
    inputType = sys.argv[3]

    if inputType == '-h':
        url = sys.argv[4]
    elif inputType == '-f':
        myfile = sys.argv[4]
        fileName, fileExt = os.path.splitext(myfile)
        fileExt = fileExt.lower()

    svgFile = sys.argv[6]

```

Ce bout de code montre l'ordre des arguments passer en ligne de commande .

La vérification des extensions et validités des fichiers passer en ligne de commande se fait comme suit :

```

        elif fileType == "xml":
            if (fileExt != '.xml'):
                print('---Erreur: Veuillez entrer un fichier XML')
            else:
                myXmlFile = xml_validation.xml_validator(myfile)
                if myXmlFile:
                    extractedXmlFile = xml_extraction.extractXmlFile(myXmlFile)
                    xml_generation.generer(extractedXmlFile)

            else:
                print("---Erreur: Veuillez specifier un type de fichier correct.")
                print("---Votre choix: "+fileType)

        elif inputType == '-h':
            url = sys.argv[4]
            print(url)
            if fileType == 'json':
                import rouguisvg
        else:
            print("---Erreur: Veuillez specifier un type d'acquisition de fichier correct.")
            print("---Votre choix: "+inputType)
    else:
        print('Erreur: Nombre d'arguments incorrect.')

```

POUR LA VALIDATION JSON

Comme on doit extraire une chaîne contenant tous les caractères du fichier, on a utilisé les méthodes **json.loads** et **file.read()** dans notre fonction **json_validator**

Bien que le module JSON convertisse les chaînes en types de données Python, les fonctions JSON sont généralement utilisées pour lire et écrire directement à partir de fichiers JSON.

Puis on récupère nos objets depuis notre fichier: **json.load()**

Comme le module json fait partie de la librairie standard de python, on l'a importé pour pouvoir l'utiliser: **import json** .

Sous **Python**, le module **json** permet de créer et de lire des données au format **json**.

La fonction **loads** permet de prendre en charge une chaîne de caractères au format **json**.

```
import json

# DEBUT Fonction de validation du fichier JSON
def json_validator(f1):
    try:
        file = open(f1)
        data = file.read()
        json.loads(data)
        return True
    except ValueError as error:
        print("Le fichier JSON n'est pas valide: %s" % error)
        return False

# FIN Fonction de validation du fichier JSON
json_validator("f1.json")
#On passe en paramètre notre fichier à la fonction
```

VALIDATION XML

Ce bout de code permet de faire la validation et l'extraction du fichier XML entré par l'utilisateur .

```
import xml_extraction
import xml.etree.ElementTree as ET
#Projet_SGBD/XJ_Convertor.py

# analyse du fichier xml
def parsefile(file):
    return ET.parse(file)

#validation du fichier xml
def xml_validator(file):
    try:
        parsedFile = parsefile(file)
        return parsedFile.getroot()
    except Exception as e:
        print("---Erreur: le fichier xml %s n'a pas une bonne syntaxe" % file)
        print("voilà c'est ça l'erreur %s" % e)
        return False
```

```
if __name__ == "__main__":
    myXmlFile = xml_validator("fichier.xml")
    if myXmlFile!=False:
        extractedXmlFile = xml_extraction.extractXmlFile(myXmlFile)
        print(extractedXmlFile)
```

EXTRACTION XML

Ce bout de code définit la fonction qui permet de faire l'extraction du fichier xml, elle s'appelle dans le fichier contenant le code de validation xml.

```
def extractXmlFile(myXmlFile):
    diagramme= myXmlFile.getchildren()
    data = list()

    for entite in diagramme:
        entite_children = entite.getchildren()
        if entite.tag=="entite":
            for entite_child in entite_children:
                canAddIt = False

                if entite_child.tag=="nom":
                    row = dict()
                    entiteName = entite_child.text
                if entite_child.tag=="attribut":
                    attributes = dict()
                    allAttributes = entite_child.getchildren()
                    for onAttribute in allAttributes:
                        canAddIt = True
                        attributes[onAttribute.tag] = onAttribute.text
                    row[entiteName] = [attributes]
            else:
                if entite_child.tag == "associations":
                    canAddIt = True
                    contentAssoc = entite_child.getchildren()
                    assocs = dict()
                    for oneContent in contentAssoc:
                        if oneContent.tag == "nom":
                            assocs["nomAssoc"] = oneContent.text
                        else:
                            if oneContent.tag == "nomAutreEntite":
                                assocs["nomAutreEntite"] = oneContent.text
                            else:
                                if oneContent.tag == "cardDeb":
                                    assocs["cardDeb"] = oneContent.text
                                else:
                                    if oneContent.tag == "cardFin":
                                        assocs["cardFin"] = oneContent.text
                    row["relations"] = {'associations':[assocs]}
                if canAddIt==True:
                    data.append(row)

    return data
```

EXTRACTION JSON

Qu'est-ce que la ressource Requests?

Requests est une bibliothèque HTTP écrite en Python.

Les requêtes vous permettront d'envoyer des requêtes HTTP l'aide de Python.

Pour notre extraction on a utilise la fonction **ExtraireDonnee (Jsodata,f1)**

```
def ExtraireDonnee(Jsodata,f1):  
    extractionDonnee = dict()  
    LEntites = []
```

Cette fonction permet d'extraire les informations qui sont dans le fichier json.Elle prend en paramètre le fichier contenant les donnée (**Jsodata**) et fichier json conserner .Elle utilise la methode du dict().

DICT()

Les opérations principales sur un dictionnaire stockent une valeur avec une clé et extraient la valeur donnée à la clé.

Pour cela on procede comme suit :

```
#Récupération de la liste des entités  
  
for i in range(len(Jsodata)):  
    keys = Jsodata[i].keys()  
    key = next(iter(keys))  
    LEntites.append(key)  
  
extractionDonnee["allEntities"] = LEntites
```

```

# Recuperation des noms des entites que l'on enregistre dans la variable "listEntites"
for i in range(len(LEntites)):
    LAttributs = []
    LAssocs = []

    # Recuperation des noms des attributs que l'on enregistre dans la variable "listAttributs"
    nbAttributs = len(Jsondata[i][LEntites[i]][0].keys())
    extractionDonnee[LEntites[i]] = dict()
    for key in f1[i][LEntites[i]][0].keys():
        LAttributs.append(key)
    extractionDonnee[LEntites[i]]["attributes"] = LAttributs
for i in range(len(LEntites)):
# Recuperations des differentes associations
    nbAssocs = len(Jsondata[i]['relations']['associations'])
for j in range(nbAssocs):
    nomAutreEntite = f1[i]['relations']['associations'][j]["nomAutreEntite"]
    nomAssoc = Jsondata[i]['relations']['associations'][j]["nomAssoc"]
return extractionDonnee

```

Et en fin :

```

#Recuperation des attributs d'une entité
def getAttributesEntity(data,entityName):
    return data[entityName]["attributes"]

```

Generation du fichier SVG :

Pour la generation du fichier svg on :

import svgwrite :

*Dans le module svgwrite on a utilisé la fonction **Text** pour afficher les noms entités et celui des attributs. La fonction **Rect** permet de représenter les rectangles désignant les entités.*

```

import svgwrite
import XJ_Convertor
import requests

if XJ_Convertor.inputType == '-f':
    JsonData = XJ_Convertor.parseJSON(XJ_Convertor.myfile)
elif XJ_Convertor.inputType == '-h':
    reponse = requests.get(url = XJ_Convertor.url)
    JsonData = reponse.json()

```

Pour la génération du fichier XML on a défini la fonction **generer** qui permet de convertir le fichier xml en json et on poursuit la generation de la meme façon qu'en json.

```

def generer(JsonData):
    LEntites = []
    # Recuperation de la liste des entites
    for i in range(len(JsonData)):
        keys = JsonData[i].keys()
        key = next(iter(keys))
        LEntites.append(key)
        print(key)

```

Ensuite on récupère la liste des entités avant d'initialiser notre fichier svg.

```

# Initiation du fichier SVG
document_svg = svgwrite.Drawing(filename = XJ_Convertor.svgFile,
                                debug = True)

```

On procède à la récupération des noms des entités et des noms des attributs comme on l'a fait au niveau de l'extraction .

```

if (i % 2 == 0):
    # Creation du rectangle contenant les infos de l'entité
    document_svg.add(document_svg.rect(insert = (10, i*150 + 10),
                                         size = ("150px", "130px"),
                                         stroke_width = "1",
                                         stroke = "black",
                                         fill = "rgb(251, 131, 107)"))

    entityCoords = {
        "nomEntite": LEntites[i],
        "coordX": 10,
        "coordY": i*150 + 10
    }
    entityCoordsList.append(entityCoords)

    document_svg.add(document_svg.rect(insert = (10, i*150 + 10),
                                         size = ("150px", "26px"),
                                         stroke_width = "1",
                                         stroke = "black",
                                         fill = "rgb(251, 131, 107)"))

```

```

# Affichage de la cardinalité depart
document_svg.add(document_svg.text(cardDeb,
    insert=(debutLigneX + 10, debutLigneY + 20),
    stroke='none',
    fill="rgb(15, 15, 15)",
    font_size='15px',
    font_weight="bold")
)

# Affichage de la cardinalité arrivée
document_svg.add(document_svg.text(cardFin,
    insert=(finLigneX - 40, finLigneY + 20),
    stroke='none',
    fill="rgb(15, 15, 15)",
    font_size='15px',
    font_weight="bold")
)

# Affichage du nom de l'association
document_svg.add(document_svg.text(nomAssoc,
    insert=((finLigneX - debutLigneX) / 2 + debutLigneX - 30, (finLigneY - debutLigneY) / 2 + debutLigneY - 30),
    stroke='none',
    fill="rgb(15, 15, 15)",
    font_size='15px',
    font_weight="bold")
)

```