rsyncFTP

1.1

Authors : Vincent Reynaert, Nicolas Sobczak, Julien Vermeil

Contents

Chapter 1

Présentation

rsyncFTP est un programme permettant de créer un dossier miroir sur un serveur ftp distant.

Le principe de ce programme est de mettre à jour le dossier situé sur le serveur distant en fonction de l'état du dossier situé en local une machine. Pour synchoniser ce fichier, il faut surveiller le dossier local et réaliser les actions nécessaires à la mise à jour du dossier miroir.

Ce programme a été réalisé en python. La version utilisée est la version 3.5.2.

Chapter 2

Choix d'organisation

Nous avons voulu séparer le plus possible les fonctionnalités en créant des fichiers différents:

- logger.py
- parser.py
- gestionFTP.py
- directorySupervisor.py
- main.py

2.1 logger.py

Le package logger contient la gestion du logger. C'est ici qu'on s'occupe de créer le logger.

2.2 parserRsyncFTP.py

Le package parserRsyncFTP contient la fonction de définition de gestion du parser. C'est ici que nous définissons les paramètres que nous passons en lignes de commandes.

2.3 gestionFTP.py

Le package gestionFTP contient toutes les fonctions utiles pour gérer les actions basiques avec le serveur FTP.

2.4 directorySupervisor.py

Le package directorySupervisor correspond à la supervision des dossiers. Il s'agit des fonctions du tp1 que nous avons adaptées pour rsyncFTP. Nous n'en avons repris que le coeur de façon à garder une certaine souplesse par rapport à ce que nous avions déjà réalisé. C'est à dire que nous l'avons amélioré notamment en supprimant les variables globales qui était dangereuses.

2.5 main.py

Le package main contient les fonctions principales. Nous avons chercher à le réduire à l'essentiel de façon à ce qu'il reste lisible et compréhensible au premier coup d'oeil.

Les quelques étapes que nous réalisons sont les suivantes :

- Nous définissons notre parser.
- Nous initialisons le logger.
- Nous lançons ensuite notre boucle principale qui supervise le dossier et synchornise le répertoire situé sur le serveur FTP distant.

Chapter 3

Choix techniques

La version de python utilisée est la 3.5.2. Le programme a été testé sur avec les serveurs FTP :

- *Filezilla* (version ..) pour windows
- *Proftpd* pour linux, avec son interface graphique Gadmin ProFTPD (version 0.4.2)

3.1 Synchronisation

Pour synchroniser le dossier miroir, nous avons choisi de surveiller notre dossier local et alors de mettre à jour le dossier miroir.

3.2 Paramètres

Nous avons choisi de passer en ligne de commande les paramètres suivants:

Paramètre	Type	Variable
site ftp distant	obligatoire	ftp
chemin vers le dossier local (directory path)	obligatoire	dp
chemin pour generer le fichier log (log path)	obligatoire	lp
2-uple contenant les extensions de la liste de fichiers a inclure et de la liste de fichiers a exclure	obligatoire	ie
chemin vers le fichier conf du log (gestion des handler)	optionnel	"-lc", "–logConf"
profondeur de la supervision du dossier, default = 2	optionnel	"-p", "–profondeur"
taille maximale des fichiers transferes en Mo, default = 500 Mo	optionnel	"-sf","–sizeFile"
frequence de supervision en s, default = 1 s	optionnel	"-f", "–frequence"
temps de supervision en s, default = 60 sec	optionnel	"-st", "–supervisionTime"

3.3 Fichier log

Concernant le fichier de log, si aucun chemin pour enregistrer le fichier n'est précisé, nous utilisons le fichier .conf qui nous définit des handlers proprement et enregistre le fichier log dans le répertoire du projet. Si un chemin est précisé, nous n'utilisons pas le fichier log. Nous créons le logger dans le code du fichier logger.py. Le fichier rsyncFTP.log est alors enregistré dans le répertoire précisé par le chemin entré en ligne de commande.

3.4 Librairie gestionFTP

Dans la librairie *gestionFTP*, nous avons choisi de définir des fonctions permettant de réaliser des actions basiques telles que :

- créer un fichier sur le seveur ftp
- transferer un fichier vers le seveur ftp
- effacer un fichier sur le seveur ftp
- créer un dossier sur le seveur ftp
- transferer un dossier vers le seveur ftp
- effacer un dossier sur le seveur ftp

Nous pouvons alors executer ces actions pour réaliser des fonctions plus complexes.

Nous avons rencontré une difficulté concernant la fonction qui supprime un dossier. Elle est un peu similaire à celle qui copie un dossier à ceci près qu'elle que l'on rencontre un probleme lorsque l'on veut supprimer un dossier ou un fichier car il n'existe plus en local. On ne peut donc pas vérifier si ce qu'on veut supprimer sur le serveur ftp est un fichier ou un dossier. En outre, pour supprimer un dossier, nous devons en supprimer tout son contenu. Il nous a donc fallu créer une fonction récursive pour vider les dossiers avant de les supprimer.

3.5 Librairie directorySupervisor

Pour pouvoir superviser un dossier, nous utilisons l'organisation des répertoire en "arbre". Nous avons choisi de ne pas créer de classe arbre mais plutôt de matérialiser un arbre avec une liste.

Nous créons donc d'abord l'arbre du dossier à chaque fois qu'il s'écoule une période correspondant à la fréquence de supervision. Puis nous comparons cette arbre avec l'arbre précédemment créé. Si on observe une modification, on l'écrit dans le fichier de log et on va signaler le type de modification effectuée afin de pouvoir mettre à jour le dossier situé sur le serveur distant.

Chapter 4

Namespace Index

4.1 Packages

Here are the packages with brief descriptions (if available):

-

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 directorySupervisor Namespace Reference

Functions

- def createSurveyList (tree, startinglevel, depth)
- def comparateSurveyList (oldListe, newListe)
- def logTheMADLists (logger, M, A, D)
- def loop (logger, frequence, supervisionTime, arbrePrecedent, dp)
- def monMain ()

6.1.1 Function Documentation

6.1.1.1 def directorySupervisor.comparateSurveyList (*oldListe, newListe*)

```
Fonction qui compare 2 listes :
:param oldListe: old list
:type oldListe: list
:param newListe:new list
:type newListe: list
:return: tuple of list for the modified files, list for the added files, list for the deleted files
:rtype: tuple
```

6.1.1.2 def directorySupervisor.createSurveyList (*tree, startinglevel, depth*)

```
Function which create a list of tupples form by (fileName, dateOfLastModif) corresponding to the files in the
:param tree: tree
:type tree: ???
:param startinglevel:
:type startinglevel: int
:param depth:
:type depth: int
:return listOfModifFiles: list for he deleted files
:rtype listOfModifFiles: list
```

6.1.1.3 def directorySupervisor.logTheMADLists (*logger, M, A, D*)

```
Function that log information contained in the 3 lists :
:param M: modified files
:type M: list
:param A: added files
:type A: list
:param D: deleted files
:type D: list
:param logger: logger
:type logger: log
```

6.1.1.4 def directorySupervisor.loop (*logger, frequence, supervisionTime, arbrePrecedent, dp*)

```
Fonction: si stop() => arret, sinon compareArbre()
:param logger: logger
:type logger: log
:param frequence: frequence de supervision
:type frequence: int
:param supervisionTime: temps de supervision, si -1 alors infini
:type supervisionTime: int
:param arbrePrecedent: tree arbre precedent
:type arbrePrecedent: tree ???
:param dp: chemin du dossier
:type dp: str
:return: 1
:rtype: int
```

6.1.1.5 def directorySupervisor.monMain ( )

6.2 gestionFTP Namespace Reference

Functions

- def connectionAuServeurFTP (host, user, password)
- def deconnexionAuServeur (connect)
- def affichageFTP (ftp)
- def envoyerUnFichier (fichier_chemin, fichier_nom, ftp)
- def etatConnexion (ftp)
- def effacerFichier (ftp, fichier)
- def creerDossier (ftp, dossier)
- def supprimerDossier (ftp, dossier)
- def lister (ftp)
- def copierContenuDossier (ftp, chemin, nom_dossier)
- def pushAuServeurFTP ()
- def monMain ()

6.2.1 Function Documentation

6.2.1.1 def gestionFTP.affichageFTP (*ftp*)

```
Fonction qui affiche ???
:param ftp: ???
:type ftp: ???
```

6.2.1.2 def gestionFTP.connectionAuServeurFTP (*host, user, password*)

```
Fonction qui initialise la connection au serveur FTP distant
:param host: host name
:type host: str
:param user: user id
:type user: str
:param password: password
:type password: str
```

6.2.1.3 def gestionFTP.copierContenuDossier (*ftp, chemin, nom_dossier*)

```
Fonction qui ???
:param ftp: ???
:type ftp: ???
:param chemin: ???
:type chemin: str
:param nom_dossier: ???
:type nom_dossier: str
```

6.2.1.4 def gestionFTP.creerDossier (*ftp, dossier*)

```
Fonction qui cree un dossier
:param ftp: ???
:type ftp: ???
:param dossier: ???
:type dossier: ???
```

6.2.1.5 def gestionFTP.deconnexionAuServeur (*connect*)

```
Fonction qui se deconnecte du serveur
:param connect: nom de la variable dans laquelle la connexion a ete declaree
:type connect: ???
```

6.2.1.6 def gestionFTP.effacerFichier (*ftp, fichier*)

```
Fonction qui ???
:param ftp: ???
:type ftp: ???
:param fichier: ???
:type fichier: ???
```

6.2.1.7 def gestionFTP.envoyerUnFichier (*fichier_chemin, fichier_nom, ftp*)

```
Fonction qui envoie un fichier
:param ftp: ???
:type ftp: ???
```

6.2.1.8 def gestionFTP.etatConnexion (*ftp*)

```
Fonction qui ???
:param ftp: ???
:type ftp: ???
```

6.2.1.9 def gestionFTP.lister (*ftp*)

```
Fonction qui ???
:param ftp: ???
:type ftp: ???
```

6.2.1.10 def gestionFTP.monMain ( )

6.2.1.11 def gestionFTP.pushAuServeurFTP ( )

```
Fonction qui push les donnees vers le serveur FTP distant
```

6.2.1.12 def gestionFTP.supprimerDossier (*ftp, dossier*)

```
Fonction qui supprime un dossier
:param ftp: ???
:type ftp: ???
:param dossier: ???
:type dossier: ???
```

6.3 logger Namespace Reference

Functions

- def initLog ()
- def monMain ()

6.3.1 Function Documentation

6.3.1.1 def logger.initLog ( )

```
log format
logging.basicConfig(datefmt='', format='%asctime', level=logging.INFO)    :param fileName: nom du fichier dans
:return: MAIN_LOGGER
:rtype: log
```

6.3.1.2 def logger.monMain ( )

6.4 main Namespace Reference

Functions

- def init (logger, args)
- def loop (logger, args)
- def monMain ()

6.4.1 Function Documentation

6.4.1.1 def main.init (*logger, args*)

```
log format
logging.basicConfig(datefmt='', format='%asctime', level=logging.INFO)
```

6.4.1.2 def main.loop (*logger, args*)

6.4.1.3 def main.monMain ( )

6.5 parser Namespace Reference

Functions

- def initVariables ()
- def monMain ()

6.5.1 Function Documentation

6.5.1.1 def parser.initVariables ( )

```
Fonction qui initialise le logger et les variables en fonction de ce que
l'utilisateur a entre.
La fonction genere une info recapitulant la liste des parametres entres.
:return: ARGS
:rtype: dict
```

6.5.1.2 def parser.monMain ( )

Chapter 7

File Documentation

7.1 src/directorySupervisor.py File Reference

Namespaces

- directorySupervisor

Functions

- def directorySupervisor.createSurveyList (tree, startinglevel, depth)
- def directorySupervisor.comparateSurveyList (oldListe, newListe)
- def directorySupervisor.logTheMADLists (logger, M, A, D)
- def directorySupervisor.loop (logger, frequence, supervisionTime, arbrePrecedent, dp)
- def directorySupervisor.monMain ()

7.2 src/gestionFTP.py File Reference

Namespaces

- gestionFTP

Functions

- def gestionFTP.connectionAuServeurFTP (host, user, password)
- def gestionFTP.deconnexionAuServeur (connect)
- def gestionFTP.affichageFTP (ftp)
- def gestionFTP.envoyerUnFichier (fichier_chemin, fichier_nom, ftp)
- def gestionFTP.etatConnexion (ftp)
- def gestionFTP.effacerFichier (ftp, fichier)
- def gestionFTP.creerDossier (ftp, dossier)
- def gestionFTP.supprimerDossier (ftp, dossier)
- def gestionFTP.lister (ftp)
- def gestionFTP.copierContenuDossier (ftp, chemin, nom_dossier)
- def gestionFTP.pushAuServeurFTP ()
- def gestionFTP.monMain ()

7.3 src/logger.py File Reference

Namespaces

- logger

Functions

- def logger.initLog ()
- def logger.monMain ()

7.4 src/main.py File Reference

Namespaces

- main

Functions

- def main.init (logger, args)
- def main.loop (logger, args)
- def main.monMain ()

7.5 src/parser.py File Reference

Namespaces

- parser

Functions

- def parser.initVariables ()
- def parser.monMain ()

Index