概述：文件监控-watchdog

python watchdog模块用于监控一个文件目录下的文件和文件夹的变动，包括文件和文件夹的增删改移。

watchdog针对不同的平台都进行了封装，不仅可以监视windows,还可以监视linux的文件系统。

一、文件系统事件基类类型定义：

watchdog.events.FileSystemEvent(event_type,src_path,is_directory=False):

- event.event_type:时间类别，moved deleted created modified
- event.src_path:触发该事件的文件或者目录的路径
- event.is_directory:该事件是否由一个目录触发

二、由watchdog.events.FileSystemEvent基类派生的子类如下：

- watchdog.events.FileDeletedEvent() 文件被删除时触发该事件
- watchdog.events.DirDeletedEvent() 目录被删除时触发该事件
- watchdog.events.DirCreatedEvent() 目录被建立时触发该事件
- watchdog.events.FileCreatedEvent() 文件被建立时触发该事件
- watchdog.events.FileModifiedEvent() 文件被修改时触发该事件
- watchdog.events.DirModifiedEvent() 目录被修改触发该事件
- watchdog.events.FileMovedEvent() 文件被移动触发该事件
- watchdog.events.DirMovedEvent() 目录被移动触发该事件

三、文件系统事件处理类：watchdog.events.FileSystemEventHandler 是事件处理器的基类，用于处理时间，使用者需要继承该类，并在子类中重写对应方法，需要重写的方法:

1.self.on_any_event(event) 任何事件发生都会首先执行该方法，该方法预设为空，dispatch()方法会先执行该方法，然后将event分派给其他方法进行处理

2.self.on_moved(event) 处理DirMovedEvent和FileMovedEvent事件，预设为空

3.self.on_created(event) 处理DirCreated和FileCreatedEvent事件 预设为空

4.self.on_deleted(event) 处理DirDeletedEvent和FileDeletedEvent事件，预设为空

5.self.on_modified(event) 处理DirModifiedEvent和FileModifiedEvent事件，预设为空

以上方法需要用到event的几个属性：

- event.is_directory:触发事件的是否为资料夹
- event.src_path:源路径
- event.dest_path：目标路径

下面是一个简单的例子：

```
from watchdog.observers import Observer
from watchdog.events import *
```

```
class FileEventHandler(FileSystemEventHandler):
    def on_any_event(self, event):
    pass

    def on_moved(self, event):
    if event.is_directory:
    print("directory moved from {0} to
{1}".format(event.src_path,event.dest_path))
    else:
    print("file moved from {0} to {1}".format(event.src_path,event.dest_path))

    def on_created(self, event):
    if event.is_directory:
    print("directory created:{0}".format(event.src_path))
    else:
    print("file created:{0}".format(event.src_path))

    def on_deleted(self, event):
    if event.is_directory:
    print("directory deleted:{0}".format(event.src_path))
    else:
    print("file deleted:{0}".format(event.src_path))

    def on_modified(self, event):
    if event.is_directory:
    print("directory modified:{0}".format(event.src_path))
    else:
    print("file modified:{0}".format(event.src_path))

if __name__ == "__main__":
    import time
    observer = Observer()
    event_handler = FileEventHandler()
    observer.schedule(event_handler, r"D:\XufiveGit\PEC\client", True)
    observer.start()
    try:
    while True:
    time.sleep(1)

    except KeyboardInterrupt:
    observer.stop()
```

补充说明：python中format方法将指定的值格式化并将它们插入到字符串的占位符中。上述代码中传入的是源路径和目标路径。

使用方法非常简单，重新定一个类FileEventHandler()继承FileSystemEventHandler类，重写四个函数，这四个函数对应文件或者目录的增删改查。

定义Observer()对象，定义FileEventHandler()对象，然后使用observer函数的对象调用schedule函数，将监控的文件目录以及FileEventHandler传入该函数，event_handler对象对于文件时间作出处理，filePath即监控的文件目录。

observer.start()启动之后，一直是处于监听的状态，除非指示退出，该程序使用time.sleep(1)保证暂停一秒（监听频率），之后发生KeyboardInterrupt异常，停止监听。

文件创建的动作其实会触发多种事件，包括FileCreatedEvent以及FileModifiedEvent事件，触发FileEventHander中重写的on_created函数以及on_modified函数，这些事需要注意的，原因在于f=open()这样的文件操作会触发FileCreatedEvent事件，执行on_created函数，文件操作f.flush()和f.close()操作会触发FileModifiedEvent事件，执行on_modified函数。

四、总结

watchdog主要采用观察者模型。主要有三个角色：observer,event_handler，被监控的文件夹。三者原本是独立的，主要通过observer.schedule函数将三者串起来，意思为observer不断检测调用平台依赖代码对监控文件夹进行变动检测，当发现改变时，通知event_handler处理