Day 13 - 提升开发效率

83次阅读

现在，我们已经把一个Web App的框架完全搭建好了，从后端的API到前端的MVVM，流程已经跑通了。

在继续工作前，注意到每次修改Python代码，都必须在命令行先Ctrl-C停止服务器，再重启，改动才能生效。

在开发阶段，每天都要修改、保存几十次代码，每次保存都手动来这么一下非常麻烦，严重地降低了我们的开发效率。有没有办法让服务器检测到代码修改后自动重新加载呢？

Django的开发环境在Debug模式下就可以做到自动重新加载，如果我们编写的服务器也能实现这个功能，就能大大提升开发效率。

可惜的是，Django没把这个功能独立出来，不用Django就享受不到，怎么办？

其实Python本身提供了重新载入模块的功能，但不是所有模块都能被重新载入。另一种思路是检测www目录下的代码改动，一旦有改动，就自动重启服务器。

按照这个思路，我们可以编写一个辅助程序pymonitor.py，让它启动wsgiapp.py，并时刻监控www目录下的代码改动，有改动时，先把当前wsgiapp.py进程杀掉，再重启，就完成了服务器进程的自动重启。

要监控目录文件的变化，我们也无需自己手动定时扫描，Python的第三方库watchdog可以利用操作系统的API来监控目录文件的变化，并发送通知。我们先用easy_install安装：

```
$ easy_install watchdog
```

利用watchdog接收文件变化的通知，如果是.py文件，就自动重启wsgiapp.py进程。

利用Python自带的subprocess实现进程的启动和终止，并把输入输出重定向到当前进程的输入输出中：

```
#!/usr/bin/env python
import os, sys, time, subprocess

from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler

def log(s):
    print '[Monitor] %s' % s

class MyFileSystemEventHander(FileSystemEventHandler):
    def __init__(self, fn):
        super(MyFileSystemEventHander, self).__init__()
        self.restart = fn

    def on_any_event(self, event):
        if event.src_path.endswith('.py'):
            log('Python source file changed: %s' % event.src_path)
            self.restart()

command = ['echo', 'ok']
process = None

def kill_process():
```

```
    global process
    if process:
        log('Kill process [%s]...' % process.pid)
        process.kill()
        process.wait()
        log('Process ended with code %s.' % process.returncode)
        process = None

def start_process():
    global process, command
    log('Start process %s...' % ' '.join(command))
    process = subprocess.Popen(command, stdin=sys.stdin, stdout=sys.stdout, stderr=sys.stderr)

def restart_process():
    kill_process()
    start_process()

def start_watch(path, callback):
    observer = Observer()
    observer.schedule(MyFileSystemEventHander(restart_process), path, recursive=True)
    observer.start()
    log('Watching directory %s...' % path)
    start_process()
    try:
        while True:
            time.sleep(0.5)
    except KeyboardInterrupt:
        observer.stop()
    observer.join()

if __name__ == '__main__':
    argv = sys.argv[1:]
    if not argv:
        print('Usage: ./pymonitor your-script.py')
        exit(0)
    if argv[0]!='python':
        argv.insert(0, 'python')
    command = argv
    path = os.path.abspath('.')
    start_watch(path, None)
```

一共50行左右的代码，就实现了Debug模式的自动重新加载。用下面的命令启动服务器：

```
$ python pymonitor.py wsgiapp.py
```

或者给pymonitor.py加上可执行权限，启动服务器：

```
$ ./pymonitor.py wsgiapp.py
```

在编辑器中打开一个py文件，修改后保存，看看命令行输出，是不是自动重启了服务器：

```
$ ./pymonitor.py wsgiapp.py
[Monitor] Watching directory /Users/michael/Github/awesome-python-webapp/www...
[Monitor] Start process python wsgiapp.py...
...
INFO:root:application (/Users/michael/Github/awesome-python-webapp/www) will start at 0.0.0.0:9000...
[Monitor] Python source file changed: /Users/michael/Github/awesome-python-webapp/www/apis.py
[Monitor] Kill process [2747]...
[Monitor] Process ended with code -9.
[Monitor] Start process python wsgiapp.py...
...
INFO:root:application (/Users/michael/Github/awesome-python-webapp/www) will start at 0.0.0.0:9000...
```

现在，只要一保存代码，就可以刷新浏览器看到效果，大大提升了开发效率。