13.11 给简单脚本增加日志功能¶

问题¶

你希望在脚本和程序中将诊断信息写入日志文件。

解决方案¶

打印日志最简单方式是使用 `logging` 模块。例如：

```
import logging

def main():
    # Configure the logging system
    logging.basicConfig(
        filename='app.log',
        level=logging.ERROR
    )

    # Variables (to make the calls that follow work)
    hostname = 'www.python.org'
    item = 'spam'
    filename = 'data.csv'
    mode = 'r'

    # Example logging calls (insert into your program)
    logging.critical('Host %s unknown', hostname)
    logging.error("Couldn't find %r", item)
    logging.warning('Feature is deprecated')
    logging.info('Opening file %r, mode=%r', filename, mode)
    logging.debug('Got here')

if __name__ == '__main__':
    main()
```

上面五个日志调用（critical(), error(), warning(), info(), debug()）以降序方式表示不同的严重级别。 `basicConfig()` 的 `level` 参数是一个过滤器。 所有级别低于此级别的日志消息都会被忽略掉。 每个logging操作的参数是一个消息字符串，后面再跟一个或多个参数。 构造最终的日志消息的时候我们使用了%操作符来格式化消息字符串。

运行这个程序后，在文件 `app.log` 中的内容应该是下面这样：

```
CRITICAL:root:Host www.python.org unknown
ERROR:root:Could not find 'spam'
```

如果你想改变输出等级，你可以修改 `basicConfig()` 调用中的参数。例如：

```
logging.basicConfig(
     filename='app.log',
     level=logging.WARNING,
     format='%(levelname)s:%(asctime)s:%(message)s')
```

最后输出变成如下：

```
CRITICAL:2012-11-20 12:27:13,595:Host www.python.org unknown
ERROR:2012-11-20 12:27:13,595:Could not find 'spam'
WARNING:2012-11-20 12:27:13,595:Feature is deprecated
```

上面的日志配置都是硬编码到程序中的。如果你想使用配置文件， 可以像下面这样修改 `basicConfig()` 调用：

```
import logging
import logging.config

def main():
    # Configure the logging system
    logging.config.fileConfig('logconfig.ini')
```

```
    ...
```

创建一个下面这样的文件，名字叫 `logconfig.ini`：

```
[loggers]
keys=root

[handlers]
keys=defaultHandler

[formatters]
keys=defaultFormatter

[logger_root]
level=INFO
handlers=defaultHandler
qualname=root

[handler_defaultHandler]
class=FileHandler
formatter=defaultFormatter
args=('app.log', 'a')

[formatter_defaultFormatter]
format=%(levelname)s:%(name)s:%(message)s
```

如果你想修改配置，可以直接编辑文件logconfig.ini即可。

讨论¶

尽管对于 `logging` 模块而已有很多更高级的配置选项， 不过这里的方案对于简单的程序和脚本已经足够了。 只想在调用日志操作前先执行下basicConfig()函数方法，你的程序就能产生日志输出了。

如果你想要你的日志消息写到标准错误中，而不是日志文件中，调用 `basicConfig()` 时不传文件名参数即可。例如：

```
logging.basicConfig(level=logging.INFO)
```

`basicConfig()` 在程序中只能被执行一次。如果你稍后想改变日志配置， 就需要先获取 `root logger`，然后直接修改它。例如：

```
logging.getLogger().level = logging.DEBUG
```

需要强调的是本节只是演示了 `logging` 模块的一些基本用法。 它可以做更多更高级的定制。 关于日志定制化一个很好的资源是 Logging Cookbook