链滴

# Flask 实战：3、启动开发服务器

作者：zhaolixiang
原文链接：https://ld246.com/article/1568557561217
来源网站：链滴

Flask内置了一个简单的开发服务器（由依赖包Werkzeug提供），足够在开发和测试阶段使用。

注意

在生产环境需要使用性能够好的生产服务器，以提升安全和性能，具体在本书第三部分会进行介绍。

# 1、Run，Flask，Run!

Flask通过依赖包Click内置了一个CLI（Command Line Interface，命令行交互界面）系统。当我们装Flask后，会自动添加一个flask命令脚本，我们可以通过flask命令执行内置命令、扩展提供的命令是我们自己定义的命令。其中，flask run命令用来启动内置的开发服务器：

```
$ flask run
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
    * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

注意

确保执行命令前激活了虚拟环境（pipenv shell），否则需要使用pipenv run flask run命令启动开发务器。后面将不再提示。

你可以执行flask --help查看所有可用的命令。

提示

如果执行flask run命令后显示命令未找到提示（command not found）或其他错误，可以尝试使用python-m flask run启动服务器，其他命令亦同。

flask run命令运行的开发服务器默认会监听http://127.0.0.1:5000/ 地址（按Crtl+C退出），并开启线程支持。当我们打开浏览器访问这个地址时，会看到网页上显示“Hello，World！”，如图1-5所。


图1-5 “Hello，World！”程序主页

提示

http://127.0.0.1 即localhost，是指向本地机的IP地址，一般用来测试。Flask默认使用5000端口，于上面的地址，你也可以使用http://localhost:5000/ 。在本书中这两者会交替使用，除了地址不同，两者没有实际区别，即域名和IP地址的映射关系。

旧的启动开发服务器的方式是使用app.run（）方法，目前已不推荐使用（deprecated）。

## 1.自动发现程序实例

一般来说，在执行flask run命令运行程序前，我们需要提供程序实例所在模块的位置。我们在上面可直接运行程序，是因为Flask会自动探测程序实例，自动探测存在下面这些规则：

- 从当前目录寻找app.py和wsgi.py模块，并从中寻找名为app或application的程序实例。
- 从环境变量FLASK_APP对应的值寻找名为app或application的程序实例。

因为我们的程序主模块命名为app.py，所以flask run命令会自动在其中寻找程序实例。如果你的程序模块是其他名称，比如hello.py，那么需要设置环境变量FLASK_APP，将包含程序实例的模块名赋值这个变量。Linux或macOS系统使用export命令：

```
$ export FLASK_APP=hello
```

在Windows系统中使用set命令：

```
> set FLASK_APP=hello
```

## 2.管理环境变量

Flask的自动发现程序实例机制还有第三条规则：如果安装了python-dotenv，那么在使用flask run其他命令时会使用它自动从.flaskenv文件和.env文件中加载环境变量。

附注

当安装了python-dotenv时，Flask在加载环境变量的优先级是：手动设置的环境变量>.env中设置的境变量>.flaskenv设置的环境变量。

除了FLASK_APP，在后面我们还会用到其他环境变量。环境变量在新创建命令行窗口或重启电脑后就除了，每次都要重设变量有些麻烦。而且如果你同时开发多个Flask程序，这个FLASK_APP就需要在同的值之间切换。为了避免频繁设置环境变量，我们可以使用python-dotenv管理项目的环境变量，先使用Pipenv将它安装到虚拟环境：

```
$ pipenv install python-dotenv
```

我们在项目根目录下分别创建两个文件：.env和.flaskenv。.flaskenv用来存储和Flask相关的公开环变量，比如FLASK_APP；而.env用来存储包含敏感信息的环境变量，比如后面我们会用来配置Email务器的账户名与密码。在.flaskenv或.env文件中，环境变量使用键值对的形式定义，每行一个，以#头的为注释，如下所示：

```
SOME_VAR=1
# 这是注释
FOO="BAR"
```

注意

.env包含敏感信息，除非是私有项目，否则绝对不能提交到Git仓库中。当你开发一个新项目时，记得它的名称添加到.gitignore文件中，这会告诉Git忽略这个文件。gitignore文件是一个名为.gitignore文本文件，它存储了项目中Git提交时的忽略文件规则清单。Python项目的.gitignore模板可以参考https://github.com/github/gitignore/blob/master/Python.gitignore 。使用PyCharm编写程序时会生一些配置文件，这些文件保存在项目根目录下的.idea目录下，关于这些文件的忽略设置可以参考https://www.gitignore.io/api/pycharm 。

## 3.使用PyCharm运行服务器

在PyCharm中，虽然我们可以使用内置的命令行窗口执行命令以启动开发服务器，但是在开发时使用yCharm内置的运行功能更加方便。在2018.1版本后的专业版添加了Flask命令行支持，在旧版本或社版中，如果要使用PyCharm运行程序，还需要进行一些设置。

首先，在PyCharm中，单击菜单栏中的Run→Edit Configurations打开运行配置窗口。图1-6中标出在PyCharm中设置一个运行配置的具体步骤序号。

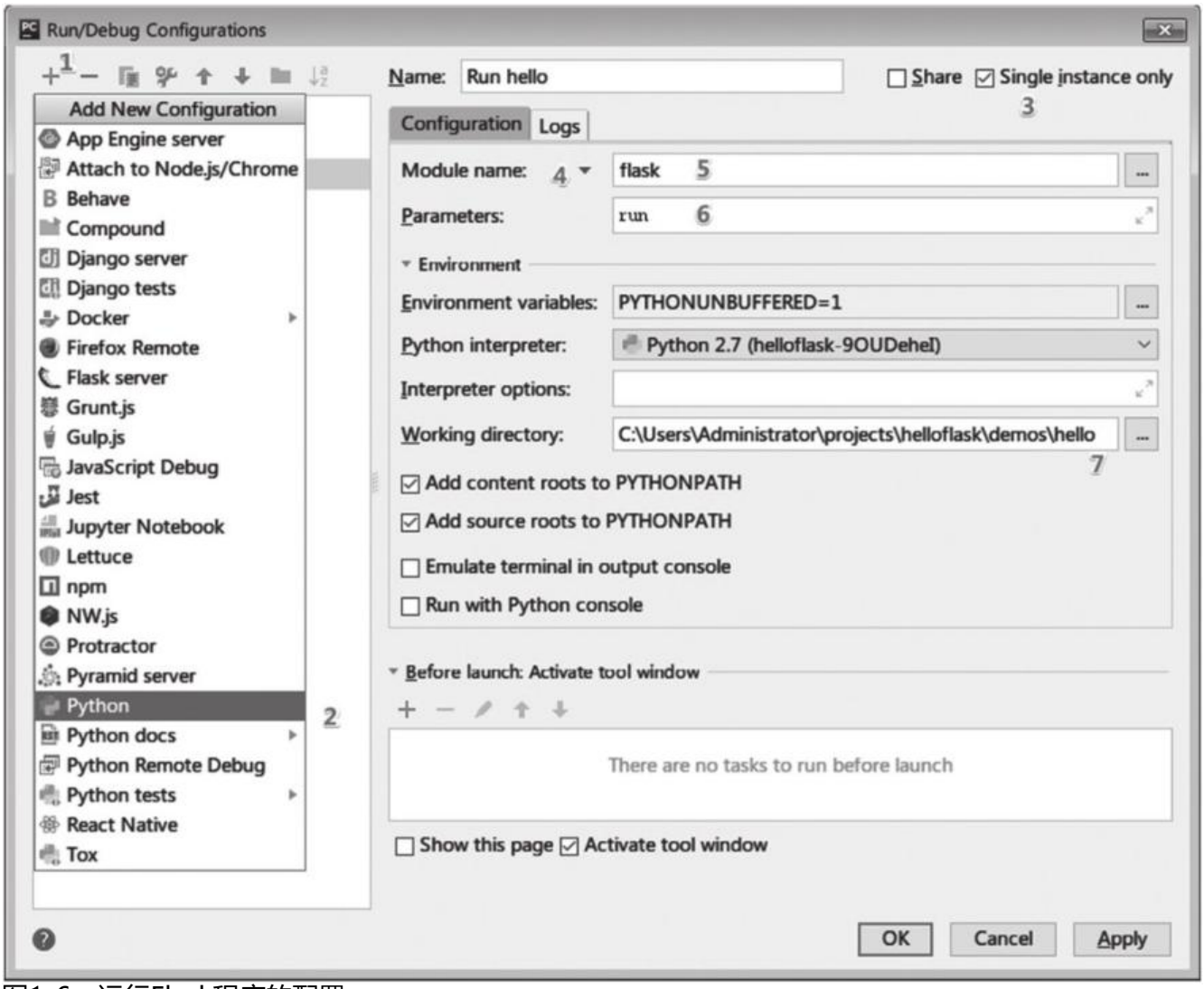

图1-6　运行Flask程序的配置

打开新建配置窗口后，具体的步骤如下所示：

- 步骤1　单击左侧的“+”符号打开下拉列表。
- 步骤2　新建一个Python类型的运行配置（如果你使用的是专业版，则可以直接选择Flask server，并在右侧的Name字段输入一个合适的名称，比如“Run hello”。
- 步骤3　勾选“Single instance only”。
- 步骤4　将第一项配置字段通过下列选项选为“Module Name”。
- 步骤5　填入模块名称flask。
- 步骤6　第二栏的“Parameters”填入要执行的命令run，你也可以附加其他启动选项。
- 步骤7　在“Working directory”字段中选择程序所在的目录作为工作目录。

提示

我们可以单击左上方的复制图标复制一份配置，然后稍加修改就可以用于其他flask命令，包括扩展提的命令，或是我们自定义的命令。

现在单击Apply或OK保存并关闭窗口。在PyCharm右上方选择我们创建的运行配置，然后单击绿色角形的运行按钮即可启动开发服务器。

注意

因为本章示例程序的模块名称为app.py，Flask会自动从中寻找程序实例，所以我们在PyCharm中的行设置可以正确启动程序。如果你不打算使用python-dotenv来管理环境变量，那么需要修改PyCham的运行配置：在Enviroment variable字段中添加环境变量FLASK_APP并设置正确的值。

# 2、更多的启动选项

## 1.使服务器外部可见

我们在上面启动的Web服务器默认是对外不可见的，可以在run命令后添加--host选项将主机地址设0.0.0.0使其对外可见：

```
$ flask run --host=0.0.0.0
```

这会让服务器监听所有外部请求。个人计算机（主机）一般没有公网IP（公有地址），所以你的程序能被局域网内的其他用户通过你的个人计算机的内网IP（私有地址）访问，比如你的内网IP为192.168191.1。当局域网内的其他用户访问http://192.168.191.1:5000 时，也会看到浏览器里显示一行“Helo，Flask！”。

提示

把程序安装在拥有公网IP的服务器上，让互联网上的所有人都可以访问是我们最后要介绍的程序部署分的内容。如果你迫切地想把你的程序分享给朋友们，可以考虑使用ngrok（https://ngrok.com/ ）Localtunnel（https://localtunnel.github.io/www/ ）等内网穿透/端口转发工具。

## 2.改变默认端口

Flask提供的Web服务器默认监听5000端口，你可以在启动时传入参数来改变它：

```
$ flask run --port=8000
```

这时服务器会监听来自8000端口的请求，程序的主页地址也相应变成了http://localhost:8000/ 。

附注

执行flask run命令时的host和port选项也可以通过环境变量FLASK_RUN_HOST和FLASK_RUN_POR设置。事实上，Flask内置的命令都可以使用这种模式定义默认选项值，即“FLASK_<COMMAND>_OPTION>”，你可以使用flask --help命令查看所有可用的命令。

# 3、设置运行环境

开发环境（development enviroment）和生产环境（production enviroment）是我们后面会频繁触到的概念。开发环境是指我们在本地编写和测试程序时的计算机环境，而生产环境与开发环境相对它指的是网站部署上线供用户访问时的服务器环境。

根据运行环境的不同，Flask程序、扩展以及其他程序会改变相应的行为和设置。为了区分程序运行环，Flask提供了一个FLASK_ENV环境变量用来设置环境，默认为production（生产）。在开发时，我可以将其设为development（开发），这会开启所有支持开发的特性。为了方便管理，我们将把环境量FLASK_ENV的值写入.flaskenv文件中：

```
FLASK_ENV=development
```

现在启动程序，你会看到下面的输出提示：

```
$ flask run
 * Environment: development
 * Debug mode: on
 * Debugger is active!
 * Debugger PIN: 202-005-064
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
111
```

在开发环境下，调试模式（Debug Mode）将被开启，这时执行flask run启动程序会自动激活Werkzug内置的调试器（debugger）和重载器（reloader），它们会为开发带来很大的帮助。

> 提示
如果你想单独控制调试模式的开关，可以通过FLASK_DEBUG环境变量设置，设为1则开启，设为0则闭，不过通常不推荐手动设置这个值。

> 注意
在生产环境中部署程序时，绝不能开启调试模式。尽管PIN码可以避免用户任意执行代码，提高攻击利用调试器的难度，但并不能确保调试器完全安全，会带来巨大的安全隐患。而且攻击者可能会通过试信息获取你的数据库结构等容易带来安全问题的信息。另一方面，调试界面显示的错误信息也会让通用户感到困惑。

### 1、调试器

Werkzeug提供的调试器非常强大，当程序出错时，我们可以在网页上看到详细的错误追踪信息，这调试错误时非常有用。运行中的调试器如图1-7所示。



调试器允许你在错误页面上执行Python代码。单击错误信息右侧的命令行图标，会弹出窗口要求输入PN码，也就是在启动服务器时命令行窗口打印出的调试器PIN码（Debugger PIN）。输入PIN码后，们可以单击错误堆栈的某个节点右侧的命令行界面图标，这会打开一个包含代码执行上下文信息的Pyton Shell，我们可以利用它来进行调试。

### 2.重载器

当我们对代码做了修改后，期望的行为是这些改动立刻作用到程序上。重载器的作用就是监测文件变，然后重新启动开发服务器。当我们修改了脚本内容并保存后，会在命令行看到下面的输出：

```
111
Detected change in '/path/to/app.py', reloading
 * Restarting with stat
111
```

默认会使用Werkzeug内置的stat重载器，它的缺点是耗电较严重，而且准确性一般。为了获得更优的体验，我们可以安装另一个用于监测文件变动的Python库Watchdog，安装后Werkzeug会自动使它来监测文件变动：

```
$ pipenv install watchdog --dev
```

因为这个包只在开发时才会用到，所以我们在安装命令后添加了一个--dev选项，这用来把这个包声为开发依赖。在Pipfile文件中，这个包会被添加到dev-packages部分。

不过，如果项目中使用了单独的CSS或JavaScript文件时，那么浏览器可能会缓存这些文件，从而导

对文件做出的修改不能立刻生效。在浏览器中，我们可以按下Crtl+F5或Shift+F5执行硬重载（hard reoad），即忽略缓存并重载（刷新）页面。

> 提示
当在一个新电脑创建运行环境时，使用pipenv install命令时需要添加额外的--dev选项才会安装dev-packages部分定义的开发依赖包。