什么是Flask-Script?

Flask-Script用来生成shell命令；为在Flask里编写额外的脚本提供了支持。

- 这包括运行一个开发服务器，一个定制的Python命令行，用于执行初始化数据库、定时任务和其他属于web应用之外的命令行任务的脚本。
- Flask-Script和Flask本身的工作方式类似。只需要定义和添加能从命令行中被Manager实例调用的命令即可。

为什么使用Flask-Script?

Flask的开发Web服务器支持很多启动设置选项，但只能在脚本中作为参数传给app.run()函数。这种方式很不方便，传递设置选项的理想方式是使用命令行参数。

- Flask-Scrip就是这么一个Flask扩展，为Flask程序**添加一个命令行解析器。**
- Flask-Script自带了一组**常用选项**，而且还**支持自定义命令。**

安装Flask-Script?

```
pip install flask_script
```

如何配置Flask-Script?

创建文件**manage.py**作为项目的入口文件。

- 无需把所有的命令都放在同一个文件里，例如，在一个大型项目中，可以把相关联的命令放在**不同**的文件里。

```
# **** manage.py文件

from flask_script import Manager
app = Flask(__name__)
# Manager类将追踪所有的在命令行中调用的命令和处理过程的调用运行情况;
# configure your app
manager = Manager(app)

if __name__ == "__main__":
    # 将启动Manger实例接收命令行中的命令。
    manager.run()
```

- 实现功能

```
python manage.py
python manage.py  runserver
python manage.py  runserver  -h
python manage.py  runserver  -h '0.0.0.0' -p 8089
```

添加自定义命令的3种方式:

- 网站参考: https://flask-script.readthedocs.io/en/latest/
    - 使用command装饰器
    - 定义Command的子类;
    - 使用option装饰器

```
# **** manage.py文件
# 方法一:
@manager.command
def hello():
    return "hello"

# 方法二:
from flask_script import Command
class Hello(Command):
    "prints hello world"

    def run(self):
        return "hello world"
manager.add_command('hello', Hello())

# 方法三:
@manager.option('-n', '--name', help='Your name')
def hello(name):
    return  "hello", name
```

6. 命令行拓展开发

> Extension developers can easily create convenient sub-manager instance within their extensions to make it easy for a user to consume all the available commands of an extension.
> Here is an example how a database extension could provide (ex. database.py):

```
# 添加其他的命令到manager里面来
from app.managerUtil.database import  database_manager
# 自定义的数据库操作
manager.add_command('database', database_manager)
# flask-script集成的数据库操作
manager.add_command('db', MigrateCommand)
```