1-建立 Django 工程

1、virtualenv

你很可能想在开发中用上 virtualenv，如果你有生产环境的 shell 权限，你同样会乐于在生产环境中使用它。
virtualenv 解决了什么问题？如果你像我一样喜欢 Python，不仅会在采用 Django 的 Web 应用中用上 virtualenv，在别的项目中你也会想用上它。你拥有的项目越多，同时使用不同版本的 Python 工作的可能性也就越大，或者起码需要不同版本的 Python 库。悲惨现实是：常常会有库破坏向后兼容性，然而正经应用不采用外部库的可能微乎其微。当在你的项目中，出现两个或更多依赖性冲突时，你会怎么做？
virtualenv 拯救世界！virtualenv 为每个不同项目提供一份 Python 安装。它并没有真正安装多个 Python 副本，但是它确实提供了一种巧妙的方式来让各项目环境保持独立。让我们来看看 virtualenv 是怎么工作的。
如果你在 Mac OS X 或 Linux 下，下面两条命令可能会适用:

```
$ sudo easy_install virtualenv
```

或更好的:

```
$ sudo pip install virtualenv
```

上述的命令会在你的系统中安装 virtualenv。它甚至可能会存在于包管理器中， 如果你用的是 Ubuntu，可以尝试:

```
$ sudo apt-get install python-virtualenv
```

如果你用的是 Windows ，而且没有 easy_install 命令，那么你必须先安装这个命令。查阅 *Windows 下的 pip 和 distribute* 章节了解如何安装。之后，运行上述的命令，但是要去掉 sudo 前缀。
virtualenv 安装完毕后，你可以立即打开 shell 然后创建你自己的环境。我通常创建一个项目文件夹，并在其下创建一个 venv 文件夹

```
$ mkdir myproject
$ cd myproject
$ virtualenv venv
New python executable in venv/bin/python
Installing distribute.............done.
```

现在，无论何时你想在某个项目上工作，只需要激活相应的环境。在 OS X 和 Linux 上，执行如下操作:

```
$ . venv/bin/activate
```

下面的操作适用 Windows:

```
$ venv\scripts\activate
```

无论通过哪种方式，你现在应该已经激活了 virtualenv（注意你的 shell 提示符显示的是当前活动的环境）。

2、blogproject

万事已经具备了，让我们来建立 Django 项目工程。

Django 工程（Project）是我们项目代码的容器，例如我们博客项目中所有的代码（包括 Django 为我们自动生成的以及我们自己写的）都包含在这个工程里。其实说通俗一点就是用一个文件夹把一系列 Python 代码文件和 Django 配置文件包裹起来，这个文件夹就可以看做一个 Django 工程。我们不必亲自动手新建这个文件夹和代码文件，Django 的内置命令已经帮我们做了这些事情。运行如下命令创建工程：

```
django-admin startproject blogproject
```

再次申明一定要在虚拟环境下运行这些命令，虚拟环境的标志就是命令提示符前有 (blogproject_env) 这样的标记。以后几乎所有和 Django 有关的命令都是在虚拟环境下运行，因此在运行前一定确保先开启了虚拟环境。

我们会发现多了一个 blogproject\ 的目录，其内部的文件结构如下：

```
blogproject\
  manage.py
```

```
blogproject\
    __init__.py
    settings.py
    urls.py
    wsgi.py
```

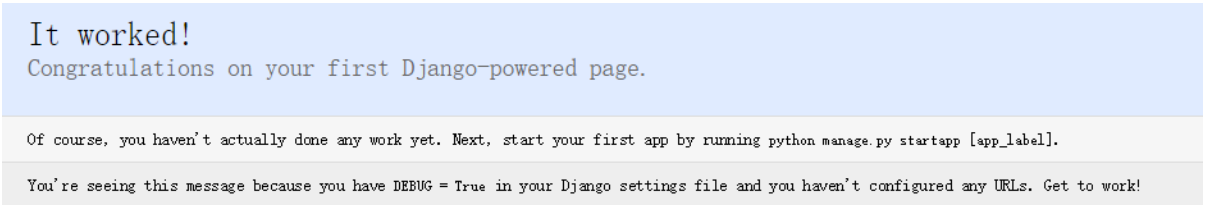
最顶层的 blogproject\ 目录是我们刚刚指定的工程目录。blogproject\ 目录下面有一个 manage.py 文件，manage 是管理的意思，顾名思义 manage.py 就是 Django 为我们生成的管理这个项目的 Python 脚本文件，以后用到时会再次介绍。与 manage.py 同级的还有一个 blogproject\ 的目录，这里面存放了一些 Django 的配置文件，例如 settings.py、urls.py 等等，以后用到时会详细介绍。

3、Hello Django

网站需要运行在一个 Web 服务器上，Django 已经为我们提供了一个用于本地开发的 Web 服务器。在命令行工具里进入到 manage.py 所在目录，即最外层的 blogproject\ 目录下。运行

```
python manage.py runserver
```

命令就可以在本机上开启一个 Web 服务器。在浏览器输入 http://127.0.0.1:8000/ ，看到如下的页面提示信息：



It worked! Django 工作了！

注意：如果在浏览器输入 http://127.0.0.1:8000/ 后显示无法访问该网站，请检查是不是浏览器代理的问题。比如开启了某些 VPN 代理服务等，将它们全部关闭即可。

这是 manage.py 的第一个用法，运行它的 `runserver` 命令开启本地开发服务器，以后我们还会遇到更多的命令。

命令栏工具下按 Ctrl + c 可以退出开发服务器（按一次没用的话连续多按几次）。重新开启则再次运行 `python manage.py runserver` 。

Django 默认的语言是英语，所以显示给我们的欢迎页面是英文的。我们在 Django 的配置文件里稍作修改，让它支持中文。用任何一个文本编辑器打开 settings.py 文件，找到如下的两行代码：

【blogproject/blogproject/settings.py】

```
## 其它配置代码...

LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'

## 其它配置代码...
```

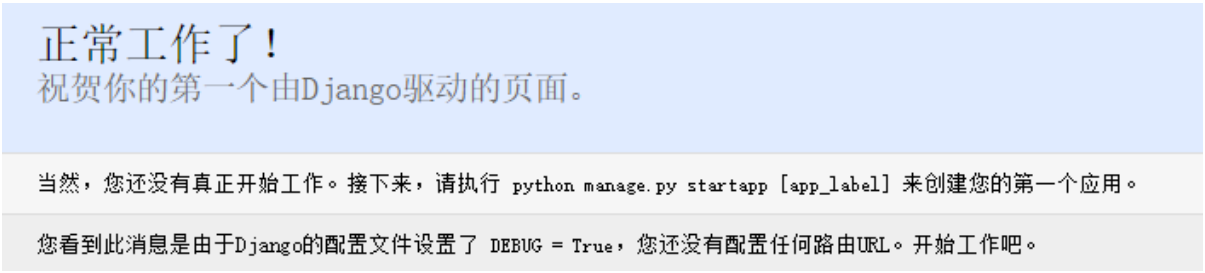
把 `LANGUAGE_CODE` 的值改为 `zh-hans`，`TIME_ZONE` 的值改为 `Asia/Shanghai`：

【blogproject/blogproject/settings.py】

```
## 其它配置代码...

# 把英文改为中文
LANGUAGE_CODE = 'zh-hans'

# 把国际时区改为中国时区
TIME_ZONE = 'Asia/Shanghai'

## 其它配置代码...
```

保存更改后关闭 settings.py 文件。

再次运行开发服务器，并在浏览器打开 http://127.0.0.1:8000/，可以看到 Django 已经支持中文了。



一切准备就绪，开始进入我们的 Django 博客开发之旅吧！