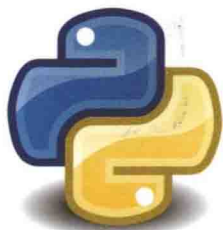


真实项目中的经验总结

行业精英们的智慧结晶



# Python

## 开发实战

团队开发环境的搭建与管理 / ticket驱动开发 / 源代码管理 (Mercurial) / PyPI包  
Jenkins持续集成 / 性能优化 (nginx与gunicorn) / GAE开发 / Django框架……

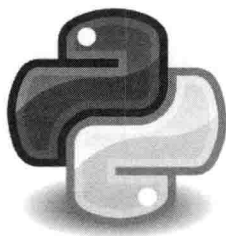
【日】BePROUD股份有限公司 著  
盛 荣 译 张靖强 审



人民邮电出版社  
POSTS & TELECOM PRESS

TURING

图灵程序设计丛书



# Python

## 开发实战

【日】BePROUD股份有限公司 著  
盛 荣 译 张靖强 审

人民邮电出版社  
北 京

## 图书在版编目 (C I P) 数据

Python开发实战 / 日本BePROUD股份有限公司著 ;  
盛荣译. — 北京 : 人民邮电出版社, 2014.6  
(图灵程序设计丛书)  
ISBN 978-7-115-32089-6

I. ①P… II. ①日… ②盛… III. ①软件工具—程序  
设计 IV. ①TP311.56

中国版本图书馆CIP数据核字(2013)第115494号

## 内 容 提 要

本书来自真正的开发现场, 是 BePROUD 公司众多极客在真实项目中的经验总结。

作者从 Python 的环境搭建讲起, 介绍了 Web 应用开发方法、任务管理, 以及评审、测试及高效部署、服务器调试等内容, 尽可能网罗了 Python 开发流程中所涉及的方方面面。在这里, Python 仅仅是一个载体, 很多知识点在非 Python 下也适用, 这也是本书最大的特色所在。

本书适合有一定基础的 Python 开发者, 以及使用 PHP 或 Ruby 进行网页开发的读者阅读。

- 
- ◆ 著 [日] BePROUD股份有限公司
  - 译 盛 荣
  - 审 张靖强
  - 责任编辑 乐 馨
  - 执行编辑 金松月
  - 责任印制 焦志炜

- ◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
- 邮编 100164 电子邮件 315@ptpress.com.cn
- 网址 <http://www.ptpress.com.cn>
- 北京天宇星印刷厂印刷

- ◆ 开本: 800×1000 1/16

印张: 24.5

字数: 379千字

印数: 1-4 000册

2014年6月第1版

2014年6月北京第1次印刷

著作权合同登记号 图字: 01-2012-9107号



---

定价: 79.00元

读者服务热线: (010)51095186转600 印装质量热线: (010)81055316

反盗版热线: (010)81055315

广告经营许可证: 京崇工商广字第 0021 号

# 版权声明

*PYTHON PROFESSIONAL PROGRAMMING*

Copyright © 2012 by BeProud

All rights reserved.

Originally published in Japan by Shuwa System CO.,LTD. Tokyo.

Chinese (in simplified character only) translation rights arranged with Shuwa System CO., LTD.  
through CREEK & RIVER Co.,LTD. Tokyo

本书中文简体字版由Shuwa System Co., LTD.授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。

# 前 言

BePROUD 股份有限公司在众多的项目开发中使用了Python编程语言。我们希望通过撰写本书，与你分享在项目实践中所积累Python使用技巧。

最近，公司的员工数量不断增加，为了让新员工快速上手，我们亦将在BePROUD工作必须掌握的基础知识写入本书中。

本书从整个工程环境的建立开始，介绍了Web应用开发方法、任务管理，以及评审、测试及高效部署、服务器调试等内容，尽可能网罗了Python开发流程中所涉及的一系列内容。如书名所示，本书更多关注项目实战技巧。

在进入正题前，我想先谈谈BePROUD公司日常工作的思维方式。

## ◎Geek/Nerd<sup>①</sup>云集的公司

BePROUD是一个Geek和Nerd众多的公司，云集了许多行业精英。

出于兴趣，这些员工经常利用自己的业余时间学习，将学到的知识付诸实践。作为Geek/Nerd，他们永远不会吝惜时间和精力。

当然，Geek/Nerd中有许多怪人，但BePROUD公司所有成员都有这样的共识：

- 尽量简化不想做的工作。
- 掌握了好的方法就要付诸实践。
- 带着好心情快乐地工作。

## ◎尽量简化不想做的工作

在工作中，重复地做一项工作是非常无聊的。我们都尽量想办法使其一劳永逸。另外，我们也不喜欢那些过程复杂且又容易失败的工作。因此我们尽可能将其简化，防止失败。

## ◎掌握了好的方法就要付诸实践

世间总有一些好办法或一些新想法，我们总是积极地在工作中去尝试。

在工作中去实践好方法往往可以减轻我们的负担。但在实践之前，我们必须先甄别出可以真正起作用的方式方法，然后再开始学习、应用。

## ◎带着好心情快乐地工作

如前所述，当掌握了好的方法，提高工作效率之后，我们就可以抱有一种好心情去享受工作

<sup>①</sup> Geek和Nerd都有痴迷某物的意思，本书指的是对于软件开发等领域极度痴迷的人。——译者注

了。比如做一个好玩的Skype机器人，在下班后找一个空闲的会议室玩LT游戏等。我们都认为，公司不仅是工作场所，也是一个让大家在做好工作的同时又能够享受它的地方。

本书的内容基于BePROUD公司员工的实践经验，因此不会强迫你去单纯地记忆复杂的知识，而是为你提供能够在实践中直接应用的内容。如果本书中介绍的知识能够对你的实际工作有帮助，我们将感到莫大的欣慰。

### ◎谢辞

感谢以下IT界同行审阅本书：寺田学 (@terapyon)、铃木孝典 (@takanory)、田村佑树 (@isoparametric)、川口耕介 (@kohsukekawa)、和田贵久 (@wadatka)、矢野勉 (@t\_yano)、涉川芳树 (@shibukawa)、西尾泰和 (@nishio)、饭野卓见 (@troter)、关根佳直、四柳刚 (@ytyng)、波田野欲一 (@tcsh)。大家在百忙之中，非常爽快地接受了审稿工作并且给出了中肯的意见，再次深表感谢。还有BePROUD公司的小田切笃 (@aodag)、菊田刚 (@nyomui)、高桥和之 (@kazyk)等，他们在繁忙的项目中抽出时间来审阅书稿，同样非常感谢他们。

最后，还要感谢那些为本书执笔却没有署名的BePROUD的各位员工。没有大家平时的斟酌切磋，本书也是无法完成的。

正是得到大家的鼎力相助，本书才得以完成。让我们一起祝愿本书能够为IT业界奉献微薄之力。

本书所有执笔者

# 导 读

## ◎本书涵盖的内容

本书分成四部分，共15章。

第一部分以个人开发为重点，介绍了如何在个人开发中构建Python开发环境（第1章），并说明了开发简单的Python应用程序的方法（第2章）。

第二部分介绍了团队开发的相关内容，依次探讨了如何准备团队开发环境（第3章）、文档（第4章）、项目管理与评审（第5章）、模块设计与单元测试（第6章）、库的发布与打包（第7章）、版本控制（第8章）、持续集成（第9章）等。这些主题都是在提高团队开发效率方面必不可少的技术与方法。

第三部分介绍了Web服务发布的环境构建与运用所必须的技术（第10章）和性能调优的方法（第11章），并详细探讨了使用Google App Engine进行Web应用构建的方法（第12章）。

第四部分先阐述了将测试引入整个开发流程从而提高项目效率这一话题（第13章），随后从Django的基础开始讲起，介绍它在实践中的使用方法（第14章）。最后讲解了提高开发速度的诀窍，如Python中实用的开发模块（第15章）。

## ◎阅读本书时需要准备

### 软件环境以及版本

- 操作系统：Ubuntu-11.10
- Python：2.7.2

### 关于操作系统

主机可以使用Windows/Mac OS/Linux等作为服务器的测试环境，运行虚拟机，在虚拟机上运行Ubuntu。

### Python的官方网站帮助链接

<http://www.python.jp/doc/2.7/><sup>①</sup>

Python官方网站的参考文档一般只介绍最基本的使用方法，有时会省略部分说明，因此建议你常备更详细的参考手册慢慢阅读。

Python官方网站的学习教程非常适合用来学习Python的安装步骤、基本语法、类以及模块。该教程中所涉及的知识，本书将不再赘述。

---

① 可以参阅<http://docs.python.org/release/2.7/>。——译者注

### Unix/Linux的一般操作命令

本书采用Ubuntu Linux作为操作系统进行阐述，因此有关Unix/Linux的基本操作均不再介绍。你如果想知道相关的操作命令，可以参考<http://linuxjm.sourceforge.jp/>。

### 关于PyPI (Python Package Index)

PyPI (<http://pypi.python.org/pypi>) 提供了众多使用pip工具进行安装的Python程序包，并且该工具还提供了程序包自动安装的相关功能<sup>①</sup>。本书用到的程序包均可在PyPI站点中找到。

### 关于敏捷开发和极限编程

本书不会针对敏捷开发<sup>②</sup>和极限编程<sup>③</sup>作个别说明。关于这两个主题有众多的书籍和网站可以供参考。

## ◎本书读者对象

本书适合如下读者：

- 希望改善个人开发环境的开发者
- 希望使团队开发更上一层楼的开发者
- 在工作中使用Python环境并想学习相关技术的人
- 新加入BePROUD公司项目的人

---

① 非常类似Perl的CPAN。——译者注

② 即agile。——译者注

③ 即eXtreme Programming。——译者注



# 目 录

## 第一部分 使用 Python 开发

01 开始使用 Python	2
01.01 Python 环境搭建	2
01.01.01 安装 apt 软件包	2
01.01.02 easy_install 与 pip	3
01.01.03 安装 virtualenv	4
01.01.04 使用不同版本的 Python	9
01.02 Mercurial 环境搭建	12
01.02.01 Mercurial 的简介	12
01.02.02 Mercurial 的安装	12
01.02.03 代码库的建立	13
01.02.04 文件的操作	14
01.03 编辑器与方便的开发工具	17
01.03.01 编辑器	17
01.03.02 Python 开发小贴士	19
01.04 本章小结	22
02 开发 Web 应用程序	23
02.01 了解 Web 应用程序	23
02.01.01 什么是 Web 应用程序	23
02.01.02 Web 应用程序与桌面应用程序的不同	24
02.01.03 Web 应用程序的机制	24
02.02 Web 应用程序开发的事前准备	27
02.02.01 关于 Flask	27
02.02.02 Flask 的安装	27
02.03 Web 应用程序的开发流程	28
02.04 明确需要开发的目标系统	28
02.04.01 留言簿应用程序的需求	28
02.04.02 明确必要的功能	29
02.04.03 明确页面的表现形式	29
02.05 页面设计	29
02.05.01 页面设计草图	30

02.05.02 设计 HTML 和 CSS	30
02.06 后台功能的开发	35
02.06.01 保存评论数据	35
02.06.02 取出保存过的评论数据	36
02.06.03 使用模板引擎显示页面	37
02.06.04 准备评论的目标 URL 地址	39
02.06.05 调整模板输出	40
02.07 功能验证	41
02.08 本章小结	42

## 第二部分 团队开发的周期和流程

03 准备团队开发环境	46
03.01 在公共服务器上建立用户、设置权限	46
03.01.01 用户与用户组的建立	46
03.01.02 sudoers	46
03.01.03 virtualenv	47
03.02 问题跟踪系统	47
03.02.01 Trac	48
03.02.02 安装 Trac	48
03.03 版本控制系统	51
03.04 有利于团队开发的工具	52
03.04.01 Skype	52
03.04.02 DropBox	53
03.04.03 Google Docs	54
03.05 本章小结	54
04 编写开发文档的准备	55
04.01 编写项目开发文档	55
04.01.01 在写文档时，我们不想做什么工作	55

04.01.02	愿意在什么样的条件下 编写文档	57	05.01.01	开始写代码……在这之 前还是先创建一个 ticket	82
04.02	Sphinx 的基础与安装	58	05.01.02	创建和 ticket 编号一致 的分支	83
04.02.01	Sphinx 的安装	58	05.01.03	让版本发布与分支对应	84
04.02.02	reStructuredText 入门	60	05.01.04	分支的合并	84
04.02.03	使用 Sphinx 编写结构化 文档的流程	61	05.01.05	整理 ticket 信息	86
04.02.04	Sphinx 的扩展	66	05.01.06	ticket 的分割	86
04.03	引入 Sphinx 后解决的问题与新的问 题	67	05.02	评审	87
04.03.01	采用纯文本, 可使用一 般的编辑器来写文档	67	05.02.01	为什么需要评审	87
04.03.02	内容与形式分离, 不关 注文档形式而专注于内 容的编写	68	05.02.02	作为被评审人: 代码评 审篇	87
04.03.03	不仅仅是 PDF, 通过一 个源可以输出多种格式	71	05.02.03	作为被评审人: 工作评 审篇	88
04.03.04	将文档分割成多个文件 编辑, 进行结构化处理	72	05.02.04	作为评审人: 代码评审 篇	89
04.03.05	通过 Mercurial 进行简单 的版本控制	73	05.02.05	作为评审人: 工作评 审篇	92
04.03.06	将 API 参考手册和对应 程序协同管理	73	05.03	本章小结	92
04.03.07	一般文档可通过 Web 浏 览器共享	76	06	模块的分割设计与单元测试	93
04.03.08	Sphinx 引入后仍需探讨 的问题	76	06.01	模块分割设计方法	93
04.04	文档集合的创建与利用	77	06.01.01	功能设计	93
04.04.01	什么是文档集合	78	06.01.02	Web 应用程序的构成 组件	94
04.04.02	项目中必不可少的文档 列表	78	06.01.03	组件设计	96
04.04.03	面向团队领导、经理	79	06.01.04	模块与包	97
04.04.04	面向设计人员	80	06.02	测试	99
04.04.05	面向开发人员	80	06.02.01	测试的种类	99
04.04.06	面向用户	81	06.02.02	编写单元测试	101
04.05	本章小结	81	06.02.03	从单元测试中去除环境 依赖	109
05	问题跟踪与评审	82	06.02.04	使用 WebTest 进行功能 测试	114
05.01	任务管理与 ticket 驱动开发	82	06.03	从测试来改善设计	118
			06.04	迈向测试执行的自动化	120
			06.04.01	测试环境的自动生成	120
			06.04.02	可以反复执行的测试 环境	121
			06.05	本章小结	121

07 打包与自动建立环境	122	07.06 本章小结	152
07.01 源代码打包与发布自动化	123	08 使用 Mercurial 管理源代码	153
07.01.01 打包	123	08.01 Mercurial 的代码库管理与配置	153
07.01.02 程序包的自动化发布	127	08.01.01 在服务器上设置 UNIX	
07.01.03 向 PyPI 注册程序包	128	用户与用户组	153
07.01.04 向 PyPI 上传程序包	128	08.01.02 设置 umask 值	154
07.01.05 书写程序包的使用说明	129	08.01.03 创建代码库	154
07.02 安装与环境关联库的介绍	131	08.01.04 配置 hgrc	155
07.02.01 Distribute	132	08.01.05 使用配置后的代码库	155
07.02.02 pip	133	08.01.06 使用 hgweb 作为简易的	
07.02.03 virtualenv	134	中央代码库	155
07.03 应用环境自动构建所必备的功能		08.02 灵活使用钩子功能	156
与工具	135	08.02.01 钩子功能的设置方法	157
07.03.01 准备独立的 Python 环		08.02.02 使用钩子脚本	157
境	135	08.02.03 钩子事件	157
07.03.02 Python 库的程序包化以		08.02.04 钩子执行的时机	159
及标准安装	136	08.02.05 编写钩子脚本	161
07.03.03 锁定应用环境的程序包		08.03 分支操作	164
版本	136	08.04 分支间的合并	165
07.03.04 整个环境可以离线安装	136	08.04.01 没有冲突的合并	165
07.03.05 重建环境时可以复用		08.04.02 用文本编辑器消除合	
缓存	137	并冲突	167
07.03.06 使用编译好的二进制		08.04.03 合并的模式与冲突的	
模块	138	种类	169
07.03.07 当 PyPI 宕机或高峰时,		08.04.04 使用图形界面合并的	
有备用服务器	138	方法	170
07.04 自动应用环境构建	140	08.05 GUI 客户端	173
07.04.01 开始	140	08.05.01 GUI 客户端的介绍	173
07.04.02 构建初始环境	140	08.05.02 GUI 客户端的优点	176
07.04.03 包含源代码的应用环		08.05.03 GUI 客户端的缺点	179
境	141	08.05.04 hgwebcommit	180
07.04.04 在部署环境中的离线		08.06 考虑实际运用而产生的 BP	
安装	143	Mercurial W/F	182
07.04.05 安装部署需要二进制		08.06.01 概要	182
编译的程序包	144	08.06.02 背景	183
07.05 公司内部共享专用库	144	08.06.03 代码库的构成	183
07.05.01 在多个项目间共享代码		08.06.04 源代码提交	184
的方法	145	08.06.05 设计师的提交操作	186
07.05.02 自动建立非公开程序包		08.06.06 分支的合并	187
的应用环境	149		

08.06.07	集成分支	188
08.07	本章小结	189
09	使用 Jenkins 持续集成	192
09.01	什么是持续集成	192
09.01.01	关于持续集成	192
09.01.02	关于 Jenkins	194
09.02	Jenkins 的安装	194
09.02.01	安装 Java	194
09.02.02	安装 Jenkins 主体程序	194
09.02.03	本章用到的 Jenkins 插件	195
09.03	执行测试代码	195
09.03.01	简单测试代码的示例	195
09.03.02	注册作业	196
09.03.03	作业执行成功与失败	198
09.04	测试结果通过报表输出	199
09.04.01	安装 nose	199
09.04.02	调用 nosetests 命令	199
09.04.03	为使用 nose 修改 Jenkins 配置	200
09.05	展示代码覆盖率报告	201
09.05.01	coverage 的安装	201
09.05.02	coverage 命令的概述： 分为“统计”和“输出” 两步	201
09.05.03	通过 nose 执行 coverage	201
09.05.04	读取代码覆盖率的报告	202
09.06	执行 Django 测试	204
09.06.01	Django 模块的安装	204
09.06.02	Django 的安装	204
09.06.03	样本代码	205
09.06.04	Jenkins 的创建	208
09.06.05	“构建后处理”部分的 配置	210
09.07	通过 Jenkins 构建文档	212
09.07.01	Sphinx 的安装	212
09.07.02	配置 Jenkins 作业的注册 信息	212

09.07.03	Sphinx 的构建在出现告 警时作业失败	213
09.07.04	查阅编译成果	214
09.07.05	使用 Task Scanner Plugin 管理 TODO 信息	215
09.07.06	Task Scanner Plugin 的 配置示例	216
09.08	进一步灵活使用 Jenkins	216
09.08.01	便捷的功能	216
09.08.02	进一步的改善	218
09.09	本章小结	219

### 第三部分 服务的公开

10	自动构建和部署环境	222
10.01	安装与配置具有依赖关系的程序 包	222
10.01.01	创建操作用户	223
10.01.02	列举必要的程序包	224
10.01.03	封闭环境的安装	226
10.01.04	应用程序的配置	228
10.01.05	环境验证	231
10.02	整理部署的操作步骤	231
10.02.01	赋予代码库访问权限	232
10.02.02	克隆源代码库	232
10.02.03	部署方法总结	232
10.03	使用 Fabric 将操作步骤自动化	233
10.03.01	什么是 Fabric	233
10.03.02	安装与环境配置	233
10.03.03	编写简单的脚本	234
10.03.04	参考步骤说明书进行自 动化	234
10.03.05	验证执行行为	238
10.03.06	添加注释	238
10.04	本章小结	239
11	改善应用程序的性能	240
11.01	Web 应用程序的性能	240

11.01.01	Web 应用遭遇大量请求 时会如何 .....	240	12.03.02	创建 SDK .....	263
11.01.02	高负载时的对策 .....	241	12.03.03	开发简单的应用程序 .....	264
11.02	留言簿应用程序的性能测试 .....	242	12.03.04	部署 .....	266
11.02.01	什么是应用程序的性能 .....	242	12.04	在 App Engine 中开发 Flask 的应用 程序 .....	267
11.02.02	ApacheBench 的安装 .....	242	12.04.01	使已有的应用程序运行 .....	267
11.02.03	使用 ApacheBench 检测 性能 .....	242	12.04.02	部署并运行 .....	271
11.03	关于 gunicorn .....	245	12.04.03	管理终端 .....	272
11.03.01	gunicorn 的安装 .....	245	12.05	调整: 改善应用程序性能 .....	274
11.03.02	使用 gunicorn 运行应用 程序 .....	245	12.05.01	根据 Appstats 进行性能 测试 .....	274
11.04	关于 nginx .....	247	12.05.02	使用多线程来处理 .....	276
11.04.01	nginx 的安装 .....	248	12.06	性能调优: 减少计费额 .....	277
11.04.02	nginx 的性能测试 .....	248	12.06.01	与计费有关的配置 .....	277
11.05	使用 nginx 和 gunicorn 共同承载应用 程序 .....	250	12.06.02	计费表 .....	277
11.05.01	gunicorn 的配置 .....	250	12.06.03	节约实例的数量 .....	279
11.05.02	nginx 的配置 .....	251	12.06.04	减少数据存储的操作 .....	280
11.05.03	测定 nginx+gunicorn 组 合的性能 .....	251	12.07	测试 .....	282
11.05.04	性能的比较 .....	252	12.07.01	在测试代码内使用 API .....	282
11.06	本章小结 .....	253	12.07.02	测试的执行 .....	284
12	Google App Engine .....	254	12.07.03	在真实服务器上测试 .....	285
12.01	关于 App Engine .....	254	12.08	本章小结 .....	285
12.01.01	什么是 App Engine .....	254	<b>第四部分 加速开发的技术</b>		
12.01.02	App Engine 的优点 .....	255	13	测试是不可分割的一部分 .....	288
12.01.03	App Engine 平台的局 限性 .....	255	13.01	认清测试现状 .....	288
12.01.04	使用案例 .....	256	13.02	在开发的各个阶段引入测试工作 .....	289
12.02	App Engine 的主要功能 .....	256	13.02.01	文档的测试 (文档评审) .....	289
12.02.01	数据存储 .....	256	13.02.02	编写测试的方法 (输入与输出) .....	292
12.02.02	自动扩展与负载分散 .....	257	13.02.03	测试的执行与测试阶段 的轮换(把什么工作做到 什么程度) .....	295
12.02.03	在 App Engine 中的 Python 运行环境 .....	259	13.03	本章小结: 对测试不要抱有恐惧 .....	298
12.03	App Engine 的开发准备工作与步骤 确认 .....	260	14	便捷地使用 Django .....	300
12.03.01	获取账号并开发程序 .....	260	14.01	什么是 Django .....	300

14.01.01	Django 的安装	300	15.02.05	编写 JSON API	336
14.01.02	Django 的架构	300	15.03	判断字符的编码	337
14.01.03	Django 的文档	304	15.04	RSS 阅读订阅的解析模块	339
14.02	让数据库的集成更加方便	304	15.04.01	导入 feedparser 模块	339
14.02.01	什么是数据库的集成	304	15.04.02	使用 feedparser 解析 RSS 阅读订阅信息	339
14.02.02	默认情况下 Django 能 做什么	304	15.05	图像处理模块	340
14.02.03	对, 就是它, 使用 South 就能够做到	305	15.05.01	安装 PIL 模块	340
14.03	使用 fixture replacement 使测试 更加方便	312	15.05.02	转换图像格式	341
14.03.01	什么是测试配置器 (fixture)	312	15.05.03	调整图像的尺寸	342
14.03.02	默认测试配置器的不便 之处	314	15.05.04	截取图像的某个部分	344
14.03.03	使用 factory_boy	315	15.05.05	图像的过滤处理	345
14.04	使调试更加方便	318	15.06	数据的加密处理模块	346
14.05	本章小结	324	15.06.01	PyCrypto 的安装	346
15	使用方便的 Python 模块	325	15.06.02	共同密钥的加密与解密	347
15.01	简化日期计算的模块	325	15.06.03	公开密钥方式的加密与 解密	347
15.01.01	日期计算的复杂之处	325	15.07	调用 Twitter 的 API 模块	351
15.01.02	引入 dateutil	327	15.07.01	导入 tweepy 模块	351
15.02	简化模型映射的模块	329	15.07.02	应用程序的登录域和 Consumer Key 的获取	351
15.02.01	必须进行模型映射的 理由	329	15.07.03	获得 Access Token	354
15.02.02	映射规则的构造与复用	330	15.07.04	调用 TwitterAPI	354
15.02.03	导入 bpmappers 模块	333	15.07.05	开发使用 Twitter 认证的 应用程序	356
15.02.04	与 Django 的联动	335	15.08	本章小结	361
			附录 A	建立 VirtualBox 环境	362
			附录 B	建立操作系统环境	366



# Part 1

## 第一部分

# 使用 Python 开发

各位读者，当你在学习一门新技术或者新的编程语言时，是不是常有无从下手的感觉？初学者在学习前，往往不知道事先应该准备什么；或刚开始准备，就马上在安装问题上栽了跟头；抑或是准备工作完成后，打算开始学习时，才发现事先应该安装的软件没有安装到位等。上述各种情况最终导致初学者在学习之初，就花费了大量的时间。

在构建 Python 的开发环境时，往往会遇到不同操作系统版本和 Python 版本，我们需要有一个兼容平滑的开发环境。此时，个人开发者尤其是自学者几乎都是参考书本或者 Web 站点等各种信息，并根据自己的需要建立开发环境。虽然这些开发环境并不完全相同，但有许多共同点。

在本书第 1 章中，我们将个人开发者建立开发环境的步骤尽可能深入浅出地展现出来。即使是初学者，也可按照里面的步骤进行操作。随后在第 2 章中，我们将会尝试实际开发一个 Web 应用。

### 本部分内容

- 第 1 章 开始使用 Python
- 第 2 章 开发 Web 应用程序

# 01 开始使用Python

本章我们将学习以下内容。我们的说明会在虚拟机环境中进行，将用来学习的主机称为“Host系统”，并将虚拟机上安装的系统称为“Guest系统”。如果你自己已经准备好了Python的开发环境，跳过本章也没有关系。

- Python 环境搭建
- Mercurial 环境搭建
- 编辑器与方便的开发工具

## 01.01 Python环境搭建

本节介绍在Ubuntu下开发Python时，需要安装的一些实用的工具包。

### NOTE

本书默认的操作系统为Ubuntu 11.10 (Server版)。该操作系统运行在Oracle VirtualBox虚拟机上。如果你还没有掌握这类环境的搭建方法，可以参考本书的附录A。初学者也请先参照附录，再阅读下面的内容。

### 01.01.01 安装apt软件包

Ubuntu使用aptitude命令进行操作系统的软件包管理<sup>①</sup>。首先我们使用下面的命令更新所有的安装包，这些安装包中包括了Python开发所必备的工具<sup>②</sup>。

#### 📦 apt软件包的更新与升级

```
$ sudo aptitude -y update
$ sudo aptitude -y upgrade
$ sudo aptitude -y install build-essential
$ sudo aptitude -y install libsqlite3-dev
$ sudo aptitude -y install libreadline6-dev
$ sudo aptitude -y install libgdbm-dev
$ sudo aptitude -y install zlibg-dev
$ sudo aptitude -y install libbz2-dev
$ sudo aptitude -y install sqlite3
$ sudo aptitude -y install tk-dev
$ sudo aptitude -y install zip
```

① aptitude类似Redhat的rpm，属于Linux“方言命令”。——译者注

② 运行命令前须先联网。——译者注



在命令中, `-y`的作用是告知安装程序: 在安装过程中, 如果遇到Y或者N的提问, 一律以Yes作为默认答案。如果你需要在安装过程中逐个确认, 可以将`-y`去掉, 再执行该命令。

`build-essential`能够在Ubuntu上编译安装Python所必备的一系列工具链(如`gcc`、`make`等)。这些基本的工具链, 在Python以及接下来提及的模块和包中, 均会作为必要的使用工具, 在这里就先行一步一起安装。<sup>①</sup>

接下来我们安装Python相关的包。

#### Python相关包的安装

```
# python-dev包的安装
$ sudo aptitude -y install python-dev

# Distribute包的安装
$ sudo chmod -R 0775 /usr/local
$ sudo chgrp -R bpbook /usr/local
$ wget http://python-distribute.org/distribute_setup.py
$ sudo python distribute_setup.py
```

`Distribute`是支持Python模块构建与导入的工具包, 虽然使用`apt`也能够安装, 但通常安装的版本较老, 不如将源代码下载下来, 进行编译安装较为妥当。关于`Distribute`包的详细信息可以参考本书第7章。

Python相关包的安装到这里就结束了, 最后我们从Ubuntu上确认一下默认安装的Python版本。

#### Python版本的确认

```
$ python -V
Python 2.7.2+
```

输入上面的命令, 就可以知道本机现在所安装的Python版本, 在Ubuntu 11.10上所安装的版本是Python 2.7.2+。

### 01.01.02 easy\_install 与 pip

Python有一个叫做PyPI (Python Package Index, <http://pypi.python.org/pypi>)的公共资源库, 它管理着Python相关的各个功能包<sup>②</sup>。任何人都可以登录该主页去下载Python相关的包资源。在使用Python开发某些软件时, 我们经常会从PyPI下载安装一些必要的功能包。

#### NOTE

PyPI的发音为/P I Pi/。

可以想象, 同Python和PyPI相类似的组合有这样几个: Perl和CPAN (<http://www.cpan.org/>)、Ruby和RubyGems (<http://rubygems.org/>)、PHP和PEAR (<http://pear.php.net/>)。

<sup>①</sup> 该命令表示更改/usr/local的目录所属用户组, 读者可根据自己情况修改。——译者注

<sup>②</sup> 熟悉Perl的读者可以将PyPI理解为Perl中的CPAN。——译者注