The Pragmatic Programmers

TURING 图灵程序设计丛书

Code in the Cloud

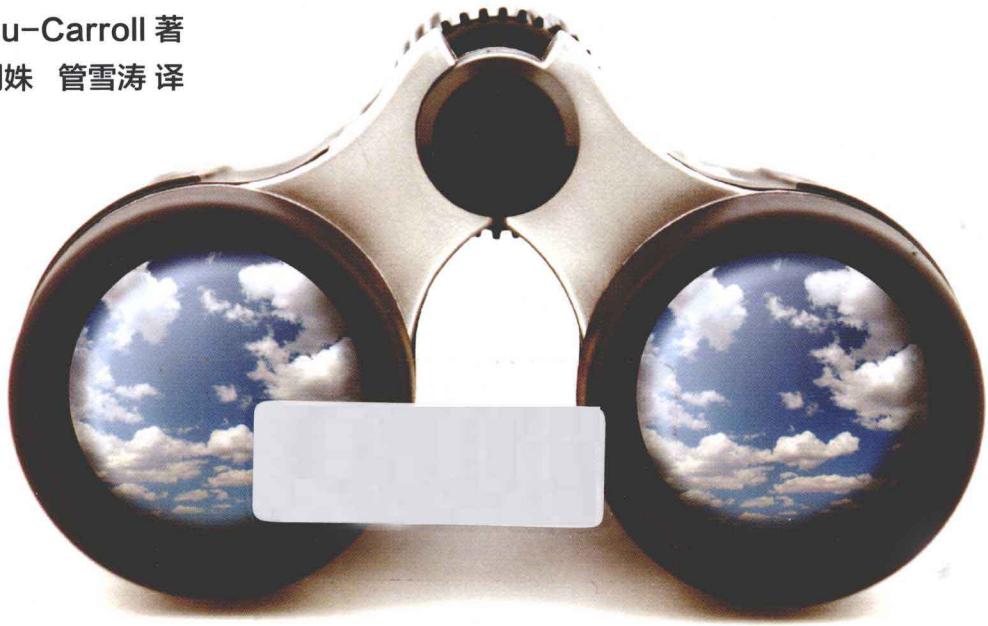
Programming Google AppEngine

云端代码

Google App Engine编程指南

[美] Mark C. Chu-Carroll 著

刘姝 管雪涛 译



人民邮电出版社

POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Code in the Cloud

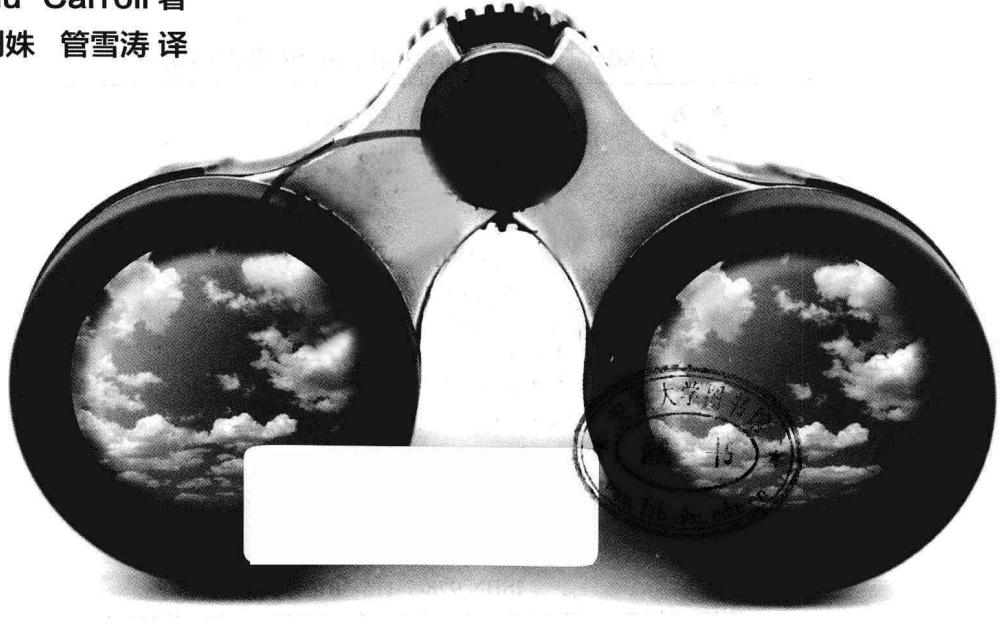
Programming Google AppEngine

云端代码

Google App Engine编程指南

[美] Mark C. Chu-Carroll 著

刘姝 管雪涛 译



人民邮电出版社

北 京

图书在版编目（CIP）数据

云端代码 : Google App Engine编程指南 / (美) 卡罗尔 (Carroll,M.C.) 著 ; 刘姝, 管雪涛译. -- 北京 : 人民邮电出版社, 2013.1
(图灵程序设计丛书)
书名原文: Code in the Cloud: Programming Google AppEngine
ISBN 978-7-115-30199-4

Ⅰ. ①云… Ⅱ. ①卡… ②刘… ③管… Ⅲ. ①网页制作工具－程序设计－指南 Ⅳ. ①TP393.092-62

中国版本图书馆CIP数据核字(2012)第289214号

内 容 提 要

本书介绍了如何将应用程序构建为服务，如何使用 App Engine 管理持久化数据，如何构建可在用户浏览器上运行的、动态的、可交互的用户界面。如何管理 Web 应用的安全性，如何用 App Engine 与云端运行的其他服务交互。

本书适合云技术开发人员、Web 程序员阅读。

图灵程序设计丛书

云端代码：Google App Engine编程指南

◆ 著 [美] Mark C. Chu-Carroll
译 刘 姝 管雪涛
责任编辑 朱 巍
执行编辑 冯 郡

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：14
字数：330千字 2013年1月第1版
印数：1－3 000册 2013年1月北京第1次印刷

著作权合同登记号 图字：01-2011-6037号

ISBN 978-7-115-30199-4

定价：45.00元

读者服务热线：(010)51095186转604 印装质量热线：(010)67129223
反盗版热线：(010)67171154

目　录

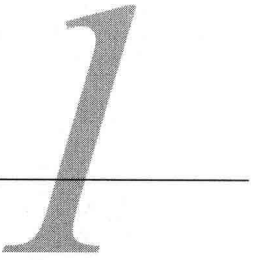
Part 1 第一部分

Google App Engine 入门

本部分内容

- 第1章 简介
- 第2章 入门

第1章 简 介



云计算是一种具有开创性的、令人兴奋的编程及使用电脑的方式。它为软件开发人员创造了巨大的机会：云计算能够为构建新型应用提供一个极棒的新平台。在这一章中，我们将了解一些基本概念：什么是云计算，何时应该使用它，为什么应该使用它，以及应用程序开发人员可以利用哪些类型的基于云的服务。

1.1 什么是云计算

在了解如何使用Google App Engine编写云程序之前，我们从最基础的开始，先弄清楚云计算指的是什么，什么是云，它与桌面计算以及老式的客户/服务器计算模式有什么不同。最重要的是要明白为什么软件开发人员需要关心云，何时需要使用云，以及应该用云来做什么。

1.1.1 云的概念

在现代互联网和万维网的世界中，数据中心分布于世界各地，每个数据中心都拥有成千上万台计算机。使用这些计算机已经成了人们的日常活动，我们通过计算机与他人聊天、发送电子邮件、玩游戏、读博客、写博客，这些活动其实是以浏览器作为客户端，去访问在服务器端运行的程序。

但是，程序实际上在哪里运行呢？数据存放在哪里？服务器在哪里？它们总归位于某个地方，放在某个数据中心，呆在世界的某个角落。用户并不知道在哪里，更重要的是，用户不用去关心，也根本没有理由去关心。用户在意的是在需要的时候要能够访问到这些程序和数据。

让我们看一个简单的例子。几年前，我开始写博客。（该博客虽然已经搬迁走了，但仍然是个很好的例子）。开始时，我使用Google的Blogger服务。每天，我会打开网络浏览器，进入http://goodmath.blogspot.com/admin，然后开始写作。写完后点击“发表”按钮，博客的内容就会呈现给我所有的读者。从我的角度来看，它就是这么工作的。我只需要网络浏览器以及URL地址，就能够写博客。

在后台，Blogger是在Google某数据中心运行的一款复杂软件。它承载了数十万的博客，并且每天都会有数百万的用户来访问这些博客。从这个角度看，显而易见，支撑Blogger的软件运行在很多台计算机上。有多少台呢？我们不知道。实际上，它甚至都可能不是一个固定的数目——

当访问用户不很多时，就不需要在很多机器上运行该软件；当越来越多的人开始用它时，就逐步需要更多的机器了。运行这个软件的机器数目是变化的。但是，从用户的角度来看——不管是博客的作者还是博客的读者，都不需要关心机器的数量这一问题。Blogger是一项服务，并且能够正常工作，这就够了。当我想写博客时，就可以进入Blogger编写；人们只要进入我的博客网页，就可以阅读它。

这就是云的基本理念：程序和数据在某个地方的计算机上，用户不需要知道也不需要关心这台计算机在哪里。

为什么将这些资源的集合称为云？云是大量微小水滴的集合。有些水滴会落在我的院子里，滋养树木和草坪，有些会流入供给我饮用水的水库里。而云本身由各地蒸发的水分形成。我只希望在院子里有足够的水分可以滋养植物，而且水库中蓄水充足让我饮用，我才不关心是哪片云带来了降雨，所有的云对我来讲都是一样的。我不关心水来自于地球的哪个地方，它们都只是水——每个水滴几乎完全相同，我看不出区别。只要有充足的水，我就心满意足了。

所以，试想世界各地的各类数据中心所在的公司都设有大量的计算机，就如同云。很多网络计算方面的最大参与者（包括Google、Amazon、Microsoft、IBM和Yahoo）都有几千台机器接入网络，运行着各类软件。每个这样的数据中心都是一片云，每个处理器、每个硬盘都是云中的水滴。在云的世界中，开发者编写程序时，不知道程序会在哪个计算机上运行。开发者不知道存储数据的硬盘在哪里，而且也不需要关心。开发者只需要清楚自己需要多少“水滴”。

1.1.2 云与开发者

云计算从根本上改变了过去计算机和软件的使用方式。传统上，如果用户想要运行某个应用程序，会去买一台计算机和一些软件，在自己家里安装好软硬件，然后运行程序。用户需要考虑该运行什么操作系统，自己安装好软件，并且需要维护计算机——跟踪软件升级、安全、备份等。

有了云计算，用户不需要再做上述工作。使用云的话，你只需要购买想要的应用程序的访问权，然后就可以在任何地方访问该应用程序。安装软件、维护程序运行所需的软硬件、保持数据安全可靠，这些问题都不需要你来考虑。在云计算环境下，你所购买的软件就是服务。如果需要比普通用户更多的存储空间，你可以从服务提供商处购买额外的存储空间。如果这样意味着需要购买和安装新的硬盘，那也由服务提供商负责。你只是从他们那里购买了存储服务，具体如何提供服务则是他们的问题。你告诉他们你需要什么，既包括实际的物理层面（“我需要1TB存储空间”），也包括略为抽象一点的服务质量层面（“我需要确保存储是事务性的，在我提交一个修改后，数据不会丢失”）。你告诉云计算服务商你有什么需求，他们就会卖给你满足这些需求的服务。

这就意味着，当开发者做云计算软件开发时，不再是购买计算机并在其上运行软件，而是将工作拆分为基本的“积木块”，然后从服务提供商那里购买这些积木块，最后根据自己想要搭建的系统将积木块组合在一起。

这些“积木块”就是开发者运行程序或者执行任务所需的资源，包括如处理时间、网络带宽、磁盘存储空间、内存等内容。作为云计算的使用者，你不需要关心这些资源位于何处。你只需知

道自己需要什么，然后从能够提供最便捷服务的提供商处购买即可。

对于开发者而言，云计算引入了更为巨大的变化。当开发者做云计算开发时，他并不是在实现一个软件，然后销售给客户，而是在为他的客户创建一个可供使用的服务。了解这其中的区别至关重要：开发者时刻要谨记是要提供给用户一种服务，而不是给他们一个要安装在他们的计算机上运行的独立应用程序。客户会根据他们要完成的任务选择服务，所以你的应用程序要根据这个任务来设计，并且必须尽可能以最灵活的方式提供相应的服务。

例如，如果开发者想为桌面计算机创建一个待办事项列表的应用程序，这是一个相当简单的过程。虽然用户界面的布局可能有很多种方式，但是创建这么一个系统的基本思路是显而易见的。开发者会创建一个用户界面——当然不需要创建多个，而且主要为单用户创建。如果为云开发待办事项列表应用程序，那么，开发者就需要有多个用户界面了：最起码，为通过桌面计算机访问这一服务的人提供一个界面，为通过手机上的移动浏览器访问这一服务的人提供一个界面。开发者可能还要提供一个开放式接口，以便其他开发人员能够用它来创建其他设备的客户端。另外，开发者需要针对多用户进行设计，因为应用程序一旦发布到云中，虽然只是单个应用程序，但却会被很多人使用。因此，开发者设计程序时要有这个准备，即使用户不会同时使用该应用程序工作，它也是一个多用户系统。

对于开发者而言，云计算最令人兴奋的一面是它的可扩展性。当开发者在云环境进行开发时，可以只编写一个供一两个人使用的简单程序，然后，甚至无需改变任何一行代码，就可以扩展到支持数百万用户。程序本身是与使用规模无关的，开发者编写的应用程序，被几十个用户使用和被上百万用户使用效果是一样的。随着用户的增加，开发者所需要的只是购买更多的资源，让程序仍然可以正常工作。开发者可以从运行在云中的一台服务器上的简单程序开始，通过增加资源而扩展到支持数百万用户。

1.1.3　云计算与客户/服务器计算

在很多方面，基于云的软件开发的基本方式和客户/服务器计算编程类似。两者都是基于同样的思想：应用程序并不真正运行在用户自己的计算机上。用户的计算机提供了访问应用程序的窗口，但是，并不直接运行应用程序本身，用户在自己的计算机上所需做的全部工作只是运行某种用户界面。真正的程序运行于其他地方被称为服务器的计算机上。之所以要使用服务器，归根结底都是因为用户的本地计算机没有运行该程序所需的资源，而在其他地方更容易获得这些资源，从而能够更便宜、更快速、更方便地运行该程序。

云开发和客户/服务器开发的最大区别在于用户知道的范围不同。在传统的客户端-服务器系统中，用户可能把一台特定的计算机作为服务器，程序就运行在该服务器上。这台服务器可能不在用户面前的办公桌上，但是用户知道它在哪里。例如，我在大学时使用过的第一台大型机是名为“Gold”的VAX 11/780，它位于希尔中心的罗格斯大学计算实验室。在亲眼看到它之前，至少一年的时间里，我几乎每天都在使用它。除Gold外，这个数据中心至少还有三十台计算机：若干台DEC 20，几台Pyramid，一台S/390，还有一群Sun服务器。但是在这些机器中，我只使用Gold。

我编写的每个程序，都专用于在Gold机上运行，而且这也是我能够运行程序的唯一的地方。

而在云中，用户却并不限定在一台特定的服务器上。用户拥有计算资源，但那是有人租给他一定数量的计算资源，这些资源位于某个地方的一些机器上，用户并不知道它们在哪里，也不知道它们是什么类型的计算机。这些机器可能是两个大型机，每个有32个处理器以及64GB的内存，也可能是64个极小的单处理器机器，每个只有2GB的内存。运行这些程序的计算机可能自身带有超大容量的硬盘，也可能是无盘工作站，需要访问专用存储服务器上的存储空间。这些对你这个云用户并不重要。你已经得到了购买的资源，而且这些资源就是你所需要的，它们的位置着实无关紧要。

1.1.4 何时用云开发

综上所述，现在我们知道了什么是云。这是一个思考计算模式的革命性方式：云是一种由服务器所组成的世界，用户可以在其上构建应用程序；云也是一种由服务所组成的世界，用户可以构建这些服务，也可以使用这些服务构建其他的东西。现在的问题是，何时该用云？

用户可以编写几乎任何一个想在云中实现的应用程序。事实上，不少人坚信，一切尽在云中，不再需要为独立的个人计算机再开发什么应用程序 。我还没有那么乐观，诚然，许多应用程序非常适合放在云里，但这并不意味着云这个平台就能完美地容纳一切。用户当然可以在云中构建任何应用程序用作服务，但是，有时这样做要比开发一个独立运行的应用程序困难得多。

在云中构建以下3种应用程序是合乎情理的。

- **协作型应用程序**

  如果用户正在构建的应用程序将被很多一起工作的团队所使用，需要进行数据共享、交流或合作，那么用户确实应该在云中构建此应用程序。协作是云的原生态。

- **服务**

  问一问“这个应用程序是干什么用的？”，如果最自然的答案听起来就像是一种服务，那么就该考虑云应用。应用程序和服务之间的区别是微妙的，我们可以把几乎所有的东西描述为服务。这里的关键问题是，什么是它最自然的描述。如果我们想要描述的是桌面版的iTunes应用程序，我们可以说：“它可以让人们管理自己的音乐收藏。”这听起来确实像是服务。但是，这一描述遗漏了iTunes桌面应用程序的关键属性：它管理用户计算机上的音乐文件，并可以通过串行电缆与iPod上的音乐文件同步。很显然，后面的描述方式意味着它是一个桌面应用程序，而不是云应用程序。

  另一方面，如果你看一下类似于eMusic的应用，你会得出不同的结论。eMusic是一个以订阅为基础的网站，用户可以浏览一个巨大的音乐库，然后每月购买一定数量的歌曲。eMusic显然是一个服务：它允许人们在成千上万的音乐曲目中进行搜索，为他们提供各种功能，如听取音乐片段、阅读各种评论、发表对所听过的音乐的评价、根据用户个人喜好推荐新音乐，并最终选择购买的东西。这显然是一种服务，放在云中合乎情理。

- **大型计算**

  你的应用程序是否需要执行海量的计算，但你却无法承担购买专用计算机的费用？如果

是这样，那么你可以通过云计算来购买服务器“农场”的时间来运行你的应用程序，这种方式经济实惠。对于像遗传学研究人员这些人来说，这真是棒极了，因为他们需要执行海量的计算，却没有资金或其他资源来为自己建立一个专用的数据中心。于是，他们可以购买商业数据中心的机时，与许多其他用户以共享的方式使用这些数据中心。

1.2　云计算编程系统

在云中进行编程的方式有多种。在真正开始编写程序之前，先快速浏览几个例子，简单了解一下几种可行方案。

1. Amazon EC2

Amazon提供了各种基于云的服务。他们的主要编程工具被称为EC2，即弹性计算云（Elastic Computing Cloud）。

事实上，EC2包括一系列相关服务。我们可以拿App Engine做个对比。App Engine提供的是一个单一的、功能极为集中的API套件，而EC2完全不管具体用什么编程API。EC2提供了数百种不同环境：用户可以使用Linux、Solaris或Windows服务器来运行其应用程序；可以使用DB2、Informix、MySQL、SQL Server或Oracle来存储其数据；可以使用Perl、Python、Ruby、Java、C++或C#来实现其代码；可以使用IBM的WebSphere或sMash、Apache JBoss、Oracle WebLogic或微软的IIS来运行其程序。根据用户喜欢的每种组合，以及计划使用的每种资源（存储空间、CPU、网络带宽）数量，成本不尽相同，既有低廉的CPU小时0.10美元和每GB带宽0.10美元，又有高端的每CPU小时0.74美元。

2. Amazon S3

Amazon还提供了另一种极为有趣的云服务，它与大多数云服务有很大不同。它是一个纯粹的存储系统，名为S3，即简单存储服务（Simple Storage Service）。S3既不运行程序，也不提供任何文件系统，更不提供任何索引，它只是一个纯粹的块存储，可以给用户分配一个存储块，存储块拥有一个唯一的标识符，然后用户就可以使用该标识符对此块进行读取和写入。

人们已经创建了使用S3存储方式的各种系统，如基于网络的文件系统、本地操作系统的文件系统、数据库系统以及表存储系统。作为以云资源为基础的范例，S3是一个非常好的例子：存储所涉及的计算与实际的数据存储本身完全分离。用户需要存储空间时，可从S3购买一定的存储空间；需要计算时，可购买EC2的资源。

S3是一个真正令人着迷的系统。它非常专一，只做存储这一件事情，并且采用了一种极其狭窄的方式。但其重要意义在于，这正是云的真谛。S3是一个绝对专一的服务，它只为用户存储数据。

S3的收费基于两个标准：用户存储数据量的大小，用户存储和读取数据所使用的网络带宽。Amazon目前的收费为，每月每GB存储空间0.15美元，带宽资源大约是，上传每GB为0.10美元，下载每GB为0.17美元。

另有相关消息称，Google提供了一个非常相似的云服务，名为“Google开发者存储”，该服务在Google云中复制了S3的基本特性。

3. IBM按需计算

IBM提供了一个云服务平台，该平台基于IBM的网络服务开发套件，采用WebSphere、DB2和Lotus协作工具。这个环境与EC2上基于IBM的环境相同，但它运行于IBM的数据中心，而不是在Amazon的数据中心。

4. Microsoft Azure

Microsoft已经开发和部署了一个名为Azure的云平台。它是一个基于Windows的平台，采用的是标准的网络服务技术（如SOAP、REST、Servlet和ASP）和微软专有API（如Silverlight）的组合。这种组合的结果是，用户可以创建一个异常强大的应用程序，非常像标准的桌面应用程序。但其缺点是，由于该应用程序与Windows平台紧密联系在一起，所以，应用的客户端主要运行在Windows平台上。虽然有其他平台的Silverlight实现，但是，这类应用往往只有在Windows平台上才是最可靠的，只有在Internet Explorer中才是功能齐全的。

因此，这就是云。既然知道了云的概念，下面我们就开始学习如何在云中构建应用程序。Google已经组建了一个很了不起的平台，名为App Engine，可以供用户构建和运行自己的云应用程序。

在本书的其余部分，我们还将详细了解一些用于构建基于云的网络应用程序的关键部件。我们将开始使用Python，用Python来学习基础知识非常棒，它可以让读者看到发生了什么事情，并且很容易快速尝试不同的方法，看看会发生什么。

我们将从基本的构建块（如HTTP、服务和处理程序）开始，贯穿读者用Python构建Google App Engine应用程序所需的各项技术的整个过程。然后，我们将了解如何利用App Engine的数据存储服务，在云中实现数据持久化存储。接下来，我们再了解如何采用HTTP、CSS和AJAX为应用程序构建用户界面。

然后，我们将暂时将内容从Python转移到Java上来。在我看来，用Java构建复杂的应用程序会更加方便。这并不是说Python不能或不应该用于高级的App Engine开发，只是我倾向于使用Java。并且App Engine提供了一个精彩绝伦的架构GWT，它从基于网络的云应用中抽象出了大部分的样板基础工作，使得用户可以把重点放在其感兴趣的部分。我们将花一些时间来学习如何使用GWT创建漂亮的用户界面，以及如何使用GWT的远程过程调用服务来实现AJAX风格的通信。

最后将介绍真实网络开发中最复杂的方面。我们将了解以下有关细节：如何使用App Engine的数据仓库服务来制作复杂系统，如何使用类似cron的机制来实现服务器端的处理和运算，以及如何将安全性和身份认证整合到用户的App Engine应用程序中。

在下一章中，我们就将正式开始学习App Engine，届时将首先介绍如何建立一个App Engine的账户，然后是如何设置用户计算机上的软件，用来构建、测试和部署采用Python编写的App Engine应用程序。

1.3　致谢

写一本书，是一个长期、艰难的过程，没有人能独自做完这件事。完成这本书，花费了很多人的很多时间。

我想要感谢以下人员。

- 技术审稿人Nick Johnson，Scott Davis，Fred Daoud，Lyle Johnson，Krishna Sankar，Dorothea Salo，感谢他们的投入和反馈。
- 本书的编辑Colleen Toporek。回想我无休止拖稿、文思枯竭，以及可怕的拼写错误……感谢她的宽容，从而保证了该书的正常进展。
- Google的App Engine团队，感谢他们构建了如此神奇的系统，为我提供了本书写作的源泉。
- 我的妻子和我绝对顽皮的孩子们，感谢她们在我埋头键盘工作的时间中对我的支持。

第2章 入 门

在这一章中，我们将初步了解并开始使用Google App Engine。我们将学习如何完成如下工作：

- 设置Google App Engine账户；
- 下载并安装Google App Engine SDK（软件开发工具包）；
- 创建一个简单的Google App Engine应用程序；
- 在本地测试应用程序；
- 在云中部署和监视Google App Engine应用程序。

这章不是本书中最令人兴奋的章节，但它是开发者能够进一步获得感兴趣内容的必经之路。当然，本章也包括一两个有趣的例子。

2.1 设置 Google App Engine 账户

为了使用Google App Engine编写云应用程序，开发者需要做的第一件事就是开通一个App Engine账户。开发者进行云开发时，需要为应用程序租用计算和存储资源。App Engine 账户为开发者提供了一组基本的免费资源，以及在需要时可以购买更多不同种类资源的机制。

创建一个Google App Engine账户是免费的。一个基本的、免费的App Engine账户可以供用户运行10个应用程序，并且具有如下特点：

- 每天6.5小时的CPU时间；
- 每天10GB的上行带宽和10GB的下行带宽；
- 1GB的数据存储空间；
- 每天可发送2000封电子邮件。

如果开发者有更多需求，可以购买各种额外资源。

要建Google App Engine账户，首先要有一个标准的Google账户。如果开发者使用了Gmail或者iGoogle，就已经有了标准账户了。如果没有，只需要访问，选择屏幕右上角的“注册”，然后点击“现在创建账户”的链接即可。

创建好Google账户之后，开发者就可以在浏览器中打开http://appengine.google.com，开始使用Google App Engine了。开发者将会看到一个标准的Google登录界面，然后用其Google用户名和

密码登录即可。第一次进行该操作时，开发者需要通过手机短信进行身份认证。为了防止垃圾邮件发送者设立App Engine账户，Google设置了使用唯一电话号码注册的机制。别想蒙混过关，一个电话号码只能设置一个App Engine账户——某号码一旦被使用，用户便不能再使用该号码创建其他账户。

CPU 时间计算

开发者每天可获得 6.5 小时的免费 CPU 时间。但是，如果开发者购买了 CPU 时间，随着工作的开展，开发者最终使用的可能远不止于此，甚至最终一天使用的 CPU 时间会超过 24 个小时。在 Google App Engine 中，开发者的应用程序不只在一台服务器上运行，而是运行在 Google 的数据中心。每个发来的请求都会被路由到集群的某台机器上。因为可以有多个用户同时访问开发者的应用系统，所以，应用程序使用了多台物理计算机的 CPU 时间。开发者要为运行其应用程序的所有计算机的 CPU 时间总和付费。这样一来，开发者一天使用的 CPU 时间可能不止 24 小时。

开发者填写完表格后，可以在浏览器中看到一个新页面，要求输入认证码。在10分钟内，开发者将会收到包含认证码的手机短信。输入认证码，就可以使用该账户了。

2.2　设置开发环境

拥有Google App Engine账户后，首先要做的就是创建一个应用程序。而自此开始，云过程的开发和通常的应用程序开发已经稍有不同了。要编写一个在自己计算机上运行的新程序，开发者只需要打开编辑器，然后输入代码即可。而编写云应用程序，开发者需要在云服务器上注册自己的应用程序，以便为其运行创建空间，并获取在该空间工作所需要的工具。

在开发者下载Google App Engine工具之前，要确保自己的计算机上已经安装了Python。Python语言现在处于不稳定状态，不断演变出被显著改写的版本。因此，日常使用中会有多个互不兼容的Python版本。对于App Engine而言，开发者需要使用Python 2.5。所以，必须确保安装的是正确的Python版本。开发者如何在不同的操作系统上安装Python用来开发App Engine服务，不是本节讨论的内容，但是，如果开发者访问Python的主页http://python.org，就可以找到最新的安装指南。

开发者还需要一个文本编辑器或者IDE来编写代码。有很多优秀的免费工具可以使用，开发者只需要挑选适合自己的那一款，并确保将它安装好即可。

当开发者安装好了编写Python程序所需的工具后，就可以登录在上一节创建的App Engine账户，点击“创建应用程序”（Create an Application）按钮，下载Google App Engine Python SDK。然后开发者将看到一个表单，显示的是应用程序的名称和描述。该表单看起来和图2-1中的表单类似。（由于App Engine经常更新，所以确切的表单可能显示出来略有不同。）

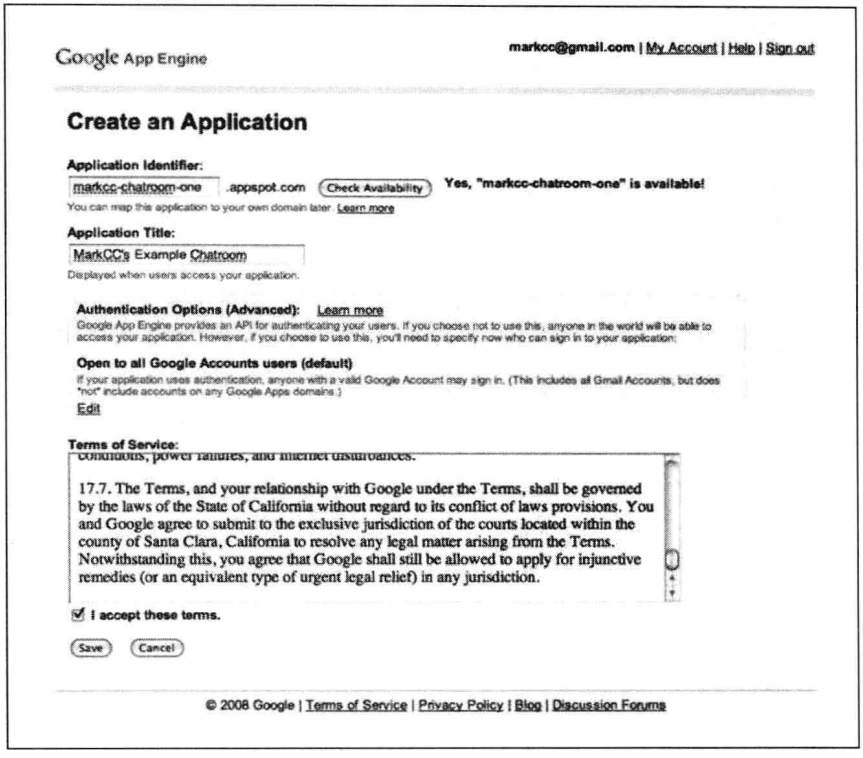


图2-1 Create an Application表单

为了创建自己的应用程序，开发者需要向Google App Engine服务提供一些信息。

❑ **应用程序标识符**

应用程序标识符（Application Identifier）是开发者的应用程序的唯一名称，以区别于任何其他App Engine用户所运行的任何一个应用程序。该名称将作为访问该应用程序的URL。由于这个属性不能被开发者修改，因此务必谨慎选择！开发者可以输入一个名称并点击“检查是否可用”（Check Availability）按钮进行检查，以确保该名称没有被其他人使用。我建议开发者为其应用程序名称选择一个私有前缀，这样做容易避免与其他人的应用程序发生名字冲突，并且在App Engine程序世界里，可以给开发者编写的一系列应用程序赋予一个通用的标识。为本书创建的所有应用程序，我都使用了`markcc`前缀。对于后文将遇到的示例应用程序，我选用的名称为`markcc-chatroom-one`，因此，该示例应用程序的URL是http://markcc-chatroom-one.appspot.com。

❑ **应用程序标题**

这是所有应用程序的用户都会看到的程序名称，并且该名称会在登录页面显示。在示例中，我使用了MarkCC's Example Chatroom（MarkCC示例聊天室）作为应用程序标题（Application Title）。开发者可以从控制面板随时修改应用程序的标题。