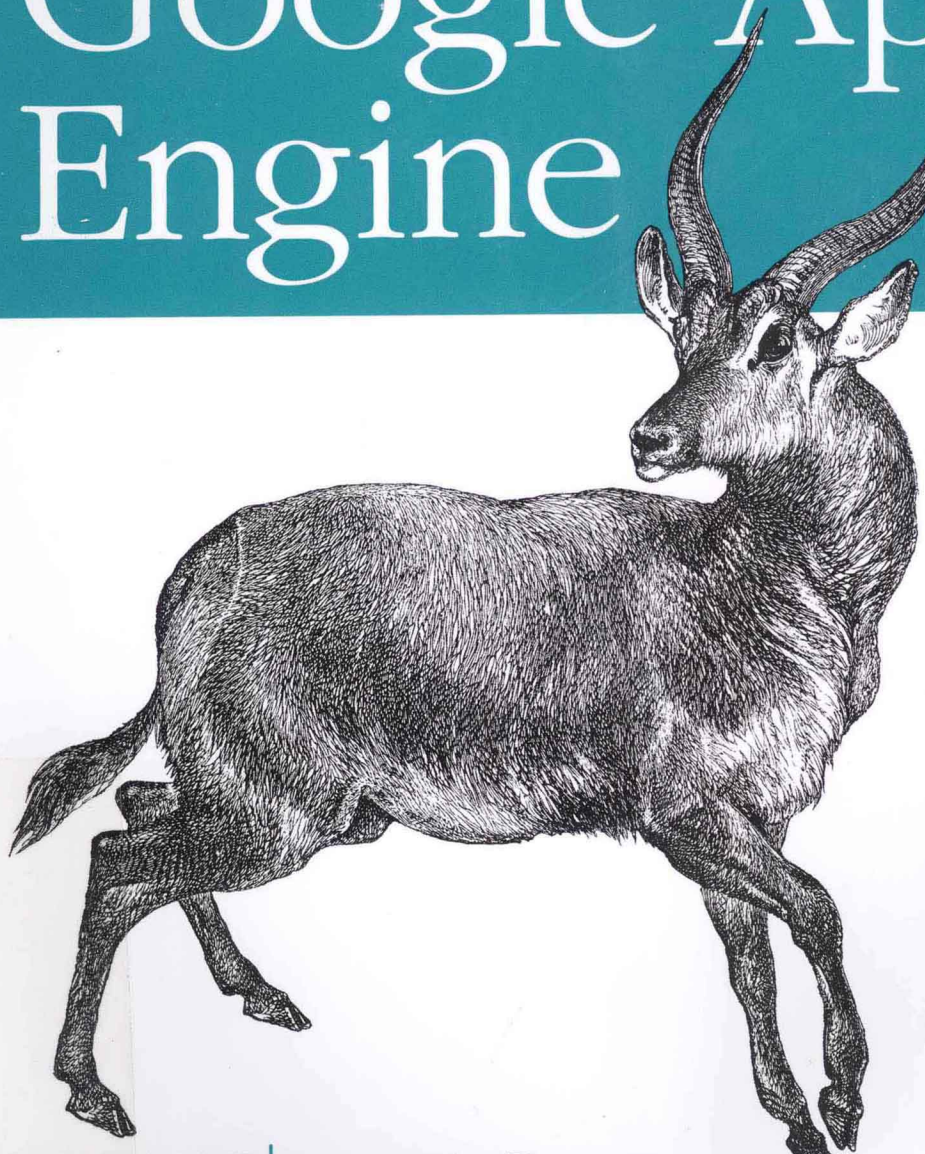
Google App Engine编程（影印版）

第二版

Programming

Google App Engine



O'REILLY® | Google Press

東南大學出版社

Dan Sanderson 著

第二版

Google App Engine编程（影印版）

Programming Google App Engine

Dan Sanderson 著

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

O'Reilly Media, Inc.授权东南大学出版社出版

南京 东南大学出版社

图书在版编目（CIP）数据

Google App Engine 编程：第 2 版：英文 /(美)桑德森(Sanderson, D.)著 . —影印本 . —南京：东南大学出版社，2013.5

书名原文：Programming Google App Engine, 2E

ISBN 978-7-5641-4180-6

I. ① G… II. ①桑… III. ①网页制作工具－程序设计－技术手册－英文 IV. ① TP393.092

中国版本图书馆 CIP 数据核字（2013）第 077529 号

江苏省版权局著作权合同登记

图字：10-2012-156 号

Google App Engine 编程 第二版（影印版）

出版发行：东南大学出版社
地　　址：南京四牌楼 2 号　　邮编：210096
出 版 人：江建中
网　　址：http://www.seupress.com
电子邮件：press@seupress.com
印　　刷：扬中市印刷有限公司
开　　本：787 毫米 × 980 毫米　16 开本
印　　张：33.5
字　　数：656 千字
版　　次：2013 年 5 月第 1 版
印　　次：2013 年 5 月第 1 次印刷
书　　号：ISBN 978-7-5641-4180-6
定　　价：69.00 元（册）

本社图书若有印装质量问题，请直接与营销部联系。电话（传真）：025-83791830

Preface

On the Internet, popularity is swift and fleeting. A mention of your website on a popular blog can bring 300,000 potential customers your way at once, all expecting to find out who you are and what you have to offer. But if you're a small company just starting out, your hardware and software aren't likely to be able to handle that kind of traffic. Chances are, you've sensibly built your site to handle the 30,000 visits per hour you're actually expecting in your first 6 months. Under heavy load, such a system would be incapable of showing even your company logo to the 270,000 others that showed up to look around. And those potential customers are not likely to come back after the traffic has subsided.

The answer is *not* to spend time and money building a system to serve millions of visitors on the first day, when those same systems are only expected to serve mere thousands per day for the subsequent months. If you delay your launch to build big, you miss the opportunity to improve your product by using feedback from your customers. Building big before allowing customers to use the product risks building something your customers don't want.

Small companies usually don't have access to large systems of servers on day one. The best they can do is to build small and hope meltdowns don't damage their reputation as they try to grow. The lucky ones find their audience, get another round of funding, and halt feature development to rebuild their product for larger capacity. The unlucky ones, well, don't.

But these days, there are other options. Large Internet companies such as Amazon.com, Google, and Microsoft are leasing parts of their high-capacity systems by using a pay-per-use model. Your website is served from those large systems, which are plenty capable of handling sudden surges in traffic and ongoing success. And since you pay only for what you use, there is no up-front investment that goes to waste when traffic is low. As your customer base grows, the costs grow proportionally.

Google App Engine, Google's application hosting service, does more than just provide access to hardware. It provides a model for building applications that grow automatically. App Engine runs your application so that each user who accesses it gets the same experience as every other user, whether there are dozens of simultaneous users or

thousands. The application uses the same large-scale services that power Google's applications for data storage and retrieval, caching, and network access. App Engine takes care of the tasks of large-scale computing, such as load balancing, data replication, and fault tolerance, automatically.

The App Engine model really kicks in at the point where a traditional system would outgrow its first database server. With such a system, adding load-balanced web servers and caching layers can get you pretty far, but when your application needs to write data to more than one place, you have a hard problem. This problem is made harder when development up to that point has relied on features of database software that were never intended for data distributed across multiple machines. By thinking about your data in terms of App Engine's model up front, you save yourself from having to rebuild the whole thing later.

Often overlooked as an advantage, App Engine's execution model helps to distribute computation as well as data. App Engine excels at allocating computing resources to small tasks quickly. This was originally designed for handling web requests from users, where generating a response for the client is the top priority. With App Engine's task queue service, medium-to-large computational tasks can be broken into chunks that are executed in parallel. Tasks are retried until they succeed, making tasks resilient in the face of service failures. The App Engine execution model encourages designs optimized for the parallelization and robustness provided by the platform.

Running on Google's infrastructure means you never have to set up a server, replace a failed hard drive, or troubleshoot a network card. And you don't have to be woken up in the middle of the night by a screaming pager because an ISP hiccup confused a service alarm. And with automatic scaling, you don't have to scramble to set up new hardware as traffic increases.

Google App Engine lets you focus on your application's functionality and user experience. You can launch early, enjoy the flood of attention, retain customers, and start improving your product with the help of your users. Your app grows with the size of your audience—up to Google-sized proportions—without having to rebuild for a new architecture. Meanwhile, your competitors are still putting out fires and configuring databases.

With this book, you will learn how to develop applications that run on Google App Engine, and how to get the most out of the scalable model. A significant portion of the book discusses the App Engine scalable datastore, which does not behave like the relational databases that have been a staple of web development for the past decade. The application model and the datastore together represent a new way of thinking about web applications that, while being almost as simple as the model we've known, requires reconsidering a few principles we often take for granted.

This book introduces the major features of App Engine, including the scalable services (such as for sending email and manipulating images), tools for deploying and managing applications, and features for integrating your application with Google Accounts and

Google Apps using your own domain name. The book also discusses techniques for optimizing your application, using task queues and offline processes, and otherwise getting the most out of Google App Engine.

Using This Book

App Engine supports three technology stacks for building web applications: Java, Python, and Go (a new programming language invented at Google). The Java technology stack lets you develop web applications by using the Java programming language (or most other languages that compile to Java bytecode or have a JVM-based interpreter) and Java web technologies such as servlets and JSPs. The Python technology stack provides a fast interpreter for the Python programming language, and is compatible with several major open source web application frameworks such as Django. The Go runtime environment compiles your Go code on the server and executes it at native CPU speeds.

This book covers concepts that apply to all three technology stacks, as well as important language-specific subjects for Java and Python. If you've already decided which language you're going to use, you probably won't be interested in information that doesn't apply to that language. This poses a challenge for a printed book: how should the text be organized so information about one technology doesn't interfere with information about the other?

Foremost, we've tried to organize the chapters by the major concepts that apply to all App Engine applications. Where necessary, chapters split into separate sections to talk about specifics for Python and Java. In cases where an example in one language illustrates a concept equally well for other languages, the example is given in Python. If Python is not your language of choice, hopefully you'll be able to glean the equivalent information from other parts of the book or from the official App Engine documentation on Google's website.

As of this writing, the Go runtime environment is released as an "experimental" feature, and the API may be changing rapidly. The language has stabilized at version 1, so if you're interested in Go, I highly recommend visiting the Go website (*http://golang.org/*) and the Go App Engine documentation (*https://developers.google.com/appengine/docs/go/overview*). We are figuring out how to best add material on Go to a future edition of this book.

The datastore is a large enough subject that it gets multiple chapters to itself. Starting with Chapter 5, datastore concepts are introduced alongside Python and Java APIs related to those concepts. Python examples use the `ext.db` data modeling library, and Java examples use the Java datastore API, both provided in the App Engine SDK. Some Java developers may prefer a higher-level data modeling library such as the Java Persistence API, which supports fewer features of the datastore but can be adapted to run

on other database solutions. We discuss data modeling libraries separately, in Chapter 9 for Python, and in Chapter 10 for Java.

This book has the following chapters:

Chapter 1, *Introducing Google App Engine*
: A high-level overview of Google App Engine and its components, tools, and major features.

Chapter 2, *Creating an Application*
: An introductory tutorial for both Python and Java, including instructions on setting up a development environment, using template engines to build web pages, setting up accounts and domain names, and deploying the application to App Engine. The tutorial application demonstrates the use of several App Engine features—Google Accounts, the datastore, and memcache—to implement a pattern common to many web applications: storing and retrieving user preferences.

Chapter 3, *Configuring an Application*
: A description of how App Engine handles incoming requests, and how to configure this behavior. This introduces App Engine's architecture, the various features of the frontend, app servers, and static file servers. The frontend routes requests to the app servers and the static file servers, and manages secure connections and Google Accounts authentication and authorization. This chapter also discusses quotas and limits, and how to raise them by setting a budget.

Chapter 4, *Request Handlers and Instances*
: A closer examination of how App Engine runs your code. App Engine routes incoming web requests to request handlers. Request handlers run in long-lived containers called instances. App Engine creates and destroys instances to accommodate the needs of your traffic. You can make better use of your instances by writing threadsafe code and enabling the multithreading feature.

Chapter 5, *Datastore Entities*
: The first of several chapters on the App Engine datastore, a scalable object data storage system with support for local transactions and two modes of consistency guarantees (strong and eventual). This chapter introduces data entities, keys and properties, and Python and Java APIs for creating, updating, and deleting entities.

Chapter 6, *Datastore Queries*
: An introduction to datastore queries and indexes, and the Python and Java APIs for queries. The App Engine datastore's query engine uses prebuilt indexes for all queries. This chapter describes the features of the query engine in detail, and how each feature uses indexes. The chapter also discusses how to define and manage indexes for your application's queries. Recent features like query cursors and projection queries are also covered.

Chapter 7, *Datastore Transactions*
: How to use transactions to keep your data consistent. The App Engine datastore uses local transactions in a scalable environment. Your app arranges its entities in

units of transactionality known as entity groups. This chapter attempts to provide a complete explanation of how the datastore updates data, and how to design your data and your app to best take advantage of these features. This edition contains updated material on the "High Replication" datastore infrastructure, and new features such as cross-group transactions.

Chapter 8, *Datastore Administration*
: Managing and evolving your app's datastore data. The Administration Console, AppCfg tools, and administrative APIs provide a myriad of views of your data, and information about your data (metadata and statistics). You can access much of this information programmatically, so you can build your own administration panels. This chapter also discusses how to use the Remote API, a proxy for building administrative tools that run on your local computer but access the live services for your app.

Chapter 9, *Data Modeling with Python*
: How to use the Python `ext.db` data modeling API to enforce invariants in your data schema. The datastore itself is schemaless, a fundamental aspect of its scalability. You can automate the enforcement of data schemas by using App Engine's data modeling interface. This chapter covers Python exclusively, though Java developers may wish to skim it for advice related to data modeling.

Chapter 10, *The Java Persistence API*
: A brief introduction to the Java Persistence API (JPA), how its concepts translate to the datastore, how to use it to model data schemas, and how using it makes your application easier to port to other environments. JPA is a Java EE standard interface. App Engine also supports another standard interface known as Java Data Objects (JDO), although JDO is not covered in this book. This chapter covers Java exclusively.

Chapter 11, *The Memory Cache*
: App Engine's memory cache service ("memcache"), and its Python and Java APIs. Aggressive caching is essential for high-performance web applications.

Chapter 12, *Large Data and the Blobstore*
: How to use App Engine's Blobstore service to accept and serve amounts of data of unlimited size—or at least, as large as your budget allows. The Blobstore can accept large file uploads from users, and serve large values as responses. An app can also create, append to, and read byte ranges from these very large values, opening up possibilities beyond serving files.

Chapter 13, *Fetching URLs and Web Resources*
: How to access other resources on the Internet via HTTP by using the URL Fetch service. This chapter covers the Python and Java interfaces, including implementations of standard URL fetching libraries. It also describes how to call the URL Fetch service asynchronously, in Python and in Java.

Chapter 14, *Sending and Receiving Email Messages*
: How to use App Engine services to send email. This chapter covers receiving email relayed by App Engine by using request handlers. It also discusses creating and processing messages by using tools in the API.

Chapter 15, *Sending and Receiving Instant Messages with XMPP*
: How to use App Engine services to send instant messages to XMPP-compatible services (such as Google Talk), and receive XMPP messages via request handlers. This chapter discusses several major XMPP activities, including managing presence.

Chapter 16, *Task Queues and Scheduled Tasks*
: How to perform work outside of user requests by using task queues. Task queues perform tasks in parallel by running your code on multiple application servers. You control the processing rate with configuration. Tasks can also be executed on a regular schedule with no user interaction.

Chapter 17, *Optimizing Service Calls*
: A summary of optimization techniques, plus detailed information on how to make asynchronous service calls, so your app can continue doing work while services process data in the background. This chapter also describes AppStats, an important tool for visualizing your app's service call behavior and finding performance bottlenecks.

Chapter 18, *The Django Web Application Framework*
: How to use the Django web application framework with the Python runtime environment. This chapter discusses setting up a project by using the Django 1.3 library included in the runtime environment, and using Django features such as component composition, URL mapping, views, and templating. With a little help from an App Engine library, you can even use Django forms with App Engine datastore models. The chapter ends with a brief discussion of django-nonrel, an open source project to connect more pieces of Django to App Engine.

Chapter 19, *Managing Request Logs*
: Everything you need to know about logging messages, browsing and searching log data in the Administration Console, and managing and downloading log data. This chapter also introduces the Logs API, which lets you manage logs programmatically within the app itself.

Chapter 20, *Deploying and Managing Applications*
: How to upload and run your app on App Engine, how to update and test an application using app versions, and how to manage and inspect the running application. This chapter also introduces other maintenance features of the Administration Console, including billing. The chapter concludes with a list of places to go for help and further reading.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This icon signifies a tip, suggestion, or general note.

This icon indicates a warning or caution.

Using Code Samples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Programming Google App Engine*, 2nd edition, by Dan Sanderson. Copyright 2013 Dan Sanderson, 978-1-449-39826-2."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online (*www.safaribooksonline.com*) is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of product mixes and pricing programs for organizations, government agencies, and individuals. Subscribers have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and dozens more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/Programming_GoogleApp_Engine*.

You can download extensive sample code and other extras from the author's website at *http://www.dansanderson.com/appengine*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

I am indebted to the App Engine team for their constant support of this book since its inception in 2008. The number of contributors to App Engine has grown too large for me to list them individually, but I'm grateful to them all for their vision, their creativity, and their work, and for letting me be a part of it. I especially want to thank Kevin Gibbs, who was App Engine's tech lead through both the first and second editions.

The first edition of the book was developed under the leadership of Paul McDonald and Pete Koomen. Ryan Barrett provided many hours of conversation and detailed technical review. Max Ross and Rafe Kaplan contributed material and extensive review to the datastore chapters. Thanks to Matthew Blain, Michael Davidson, Alex Gaysinsky, Peter McKenzie, Don Schwarz, and Jeffrey Scudder for reviewing portions of the first edition in detail, as well as Sean Lynch, Brett Slatkin, Mike Repass, and Guido van Rossum for their support. For the second edition, I want to thank Peter Magnusson, Greg D'alesandre, Tom Van Waardhuizen, Mike Aizatsky, Wesley Chun, Johan Euphrosine, Alfred Fuller, Andrew Gerrand, Sebastian Kreft, Moishe Lettvin, John Mulhausen, Robert Schuppenies, David Symonds, and Eric Willigers.

Thanks also to Steven Hines, David McLaughlin, Mike Winton, Andres Ferrate, Dan Morrill, Mark Pilgrim, Steffi Wu, Karen Wickre, Jane Penner, Jon Murchinson, Tom Stocky, Vic Gundotra, Bill Coughran, and Alan Eustace.

At O'Reilly, I'd like to thank Michael Loukides and Meghan Blanchette for giving me this opportunity and helping me see it through to the end, twice.

I dedicate this book to Google's site-reliability engineers. It is they who carry the pagers, so we don't have to. We are forever grateful.

Table of Contents