THE EXPERT'S VOICE® IN CLOUD COMPUTING

Beginning Java™ Google App Engine

Learn about the core components of the Google App Engine SDK, platform, and services for web developers using Java™ technology

Kyle Roche and Jeff Douglas

Apress®

Beginning Java™ Google App Engine

Kyle Roche
Jeff Douglas

Apress®

Beginning Java™ Google App Engine

Copyright © 2009 by Kyle Roche and Jeff Douglas

ISBN-13 (pbk): 978-1-4302-2553-9

ISBN-13 (electronic): 978-1-4302-2554-6

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please e-mail info@apress.com, or visit `http://www.apress.com`.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at `http://www.apress.com/info/bulksales`.

The source code for this book is available to readers at `http://www.apress.com`. You will need to answer questions pertaining to this book in order to successfully download the code.

There’s an Irish saying . . . no man can prosper without his woman’s leave. —KR

To Cathy, who has touched not only my heart,
but the hearts of so many that will never remember her. I love you. —JD

Contents at a Glance

Contents

Foreword

You've just picked up a book on Google App Engine. Welcome to the ground floor of a critical component in our industry's shift to cloud computing.

It's not an exaggeration to say that the development of consumer and enterprise applications has been completely transformed by the emergence of cloud computing over the past several years. First came a revolution in application delivery—the idea that applications could be delivered as a service over the Internet, without any software to install or maintain. Then came a revolution in application infrastructure—the idea that developers could consume raw computing and storage capabilities as a service, without any physical infrastructure to deploy or maintain.

Now we're seeing a revolution in application platforms—giving developers the ability to build applications using higher-level building blocks, without needing to know about the underlying physical machine. App Engine is Google's entry into this world of on-demand application development and deployment, and represents a major contribution in this shift to the cloud. Here's why App Engine is so important:

1. Development without worrying about deployment infrastructure
Most application development projects require a lot of time for planning the development and deployment stack. Which app-server container, database server, and load balancer should you use? Do you have enough licenses to deploy? Is your app going to share an existing database or do you need to spin up a new instance? How will you back up and monitor the performance of the app? Do you have enough CPU, data, and network resources to adequately scale your app? All these questions had to be answered before you could write a single line of code. Google App Engine changes all that. Google provides a complete development and deployment stack, and you can start developing with no up-front cost. Google does the heavy lifting, allowing you to focus on the specific needs of your users.

2. Single development environment, end to end
Database development, application development, and UI development have traditionally been done in completely different environments, often by completely different development teams. With App Engine's integration with Google Web Toolkit, you can download the SDK, install the Eclipse plug-in, and start to code your entire application in a single environment. You can build your UI directly in Java,

connect it to App Engine Java Data Objects, and debug everything end to end, all from within Eclipse.

3. Instant deployment, cloud scalability
Traditional application developers allocate up to one third of their total development time to deployment into a production environment. Your first App Engine app will deploy from your local development environment to Google's world-class, cloud-scale production infrastructure, all with a press of a button. And your application can scale from its first user to its millionth user with complete elasticity, literally running on the same infrastructure as the highest traffic sites on the Internet.

The implications?
Given Google App Engine's new capabilities, we've been excited to add it to the set of tools that we use at Appirio to help our enterprise customers do more with the cloud. App Engine fills a recognized gulf between the two leading cloud platforms, Force.com and Amazon Web Services. Force.com is a rich business application platform with built-in business objects that allow applications to inherit a broad swath of functionality. But some applications don't require this functionality and would benefit from having greater control and direct access to "lower levels" of the platform. At the other end of the spectrum, Amazon Web Services, in particular S3 and EC2, give application developers the power to control their own infrastructure without the headaches of hardware ownership. But many applications don't require this level of control of the infrastructure; a higher level of abstraction would make development much more efficient.

We see Google App Engine as filling the void between these two leading platforms. App Engine offers more control than you get from working in a Force.com environment. And App Engine offers abstraction over several layers of infrastructure that we'd prefer not to deal with in the applications that we build today on EC2, so, for example, we don’t have to worry about the size of the machine we spin up.

The best part is that these technologies are almost completely complementary, and toolkits exist to ease their interoperability. At an event this year, someone posed the following question: “Is the industry on the verge of a new set of platform wars? Or will all the different cloud platforms create an interwoven fabric of web applications that draw from each cloud as is convenient?" We believe firmly in the latter. After all, the real “platform war” is still against the old paradigm. Most developers out there don’t know that they don’t need to buy hardware and software anymore in order to develop and deploy world-class web applications.

But you will. Enjoy this introduction to the new world of developing on Google’s App Engine. We look forward to seeing the applications that you develop!

Ryan Nichols
V.P. Cloud Strategy, Appirio

About the Authors

Kyle Roche has been working in the cloud-computing space since 2005. Professionally, Kyle has over 10 years of experience in the enterprise software space. With deep roots in application architecture and systems management he quickly recognized cloud computing as the future trend and has since led some of the most progressive cloud-development efforts to date for companies like Salesforce.com, Starbucks, and JP Morgan Chase. Kyle is a regular speaker at industry conferences and user-group meetings and is an evangelist for cloud computing. His personal website is `http://www.kyleroche.com`.

He lives in Denver with his wife Jessica and his three children Aodhan, Avery, and Kelly.

Jeff Douglas is a highly sought-after and award-winning technologist with more than 15 years of leadership experience crafting technology solutions for companies of all sizes. His technology skills were honed during the fast and furious "dot com era," when he provided SAP development services for Fortune 500 companies including Coca-Cola, Anheuser-Busch, Disney Imagineering, Moen, and Ericsson. After years of being a lowly Java developer, in 2006 he ascended into cloud computing. He periodically writes for developer.force.com and actively tries to work the word "chartreuse" into everyday technical conversations. He speaks at industry conferences and enthusiastically blogs about cloud computing at `http://blog.jeffdouglas.com`.

Jeff resides in Sarasota, FL, with his wife Cathy and four children Scott, Tyler, Brittany, and Kira (adopted). He and his wife have been medical foster parents for over 11 years, caring for more than 75 children.

Kyle and Jeff both work for Appirio, a cloud solution provider that offers both products and professional services to help enterprises accelerate their adoption of the cloud. With over 2,500 customers, Appirio has a proven track record of implementing mission-critical solutions and developing innovative products on cloud platforms such as Salesforce.com, Google Apps, and Amazon Web Services. From offices in the U.S. and Japan, Appirio serves a wide range of companies including Avago, Hamilton Beach, Japan Post Network, Ltd, Pfizer, and Qualcomm. Appirio was founded in 2006, is the fastest growing partner of Salesforce.com and Google, and is backed by Sequoia Capital and GGV Capital.

About the Technical Reviewer

Kunal Mittal serves as an Executive Director of Technology at Sony Pictures Entertainment where he is responsible for the SOA and Identity Management programs. He provides a centralized engineering service to different lines of business and consults on the open-source technologies, content management, collaboration, and mobile strategies.

Kunal is an entrepreneur who helps startups define their technology strategy, product roadmap, and development plans. Having strong relations with several development partners worldwide, he is able to help startups and large companies build appropriate development partnerships. He generally works in an advisor or consulting CTO capacity, and serves actively in the Project Management and Technical Architect functions.

He has authored and edited several books and articles on J2EE, cloud computing, and mobile technologies. He holds a Master's degree in Software Engineering and is an instrument-rated private pilot.

Acknowledgments

This was an exciting title for Jeff and me. It's difficult to get a book together on a technology that was launched just weeks before and releases updates faster then we're drafting chapters. As this was our first printed publication we had a lot to learn about the process. It was a growing experience for us and we want to thank some of the key people who helped make this possible.

First, we'd like to thank our families who gave up a lot of their weekend and evening time to allow this project to get completed. Thanks to my wife, Jessica, and my three children: Aodhan, Avery, and Kelly. And thanks to Jeff's wife, Cathy, who went so far as to proofread all of his chapters before they were sent in, and his children: Scott, Tyler, Brittany, and Kira.

Next, we'd like to acknowledge the team led by Kelly Moritz and Steve Anglin at Apress that coordinated this entire effort. Kelly really helped us through our first printed publication. It's a difficult process and without her guidance this wouldn't have happened. Steve persevered through endless back and forth communications, refining the abstract and the concept for the book. Of course, we'd also like to acknowledge the editing team of Tom Welsh and Matthew Moodie, and our technical reviewer Kunal Mittal, for their patience with two over-eager first-time writers.

Also, thanks to the Appirio team for putting us in a position to write a book on this emerging technology. Appirio is like no other company we've encountered. Having been a part of some of the most progressive cloud-computing projects to date, we constantly get the opportunity to work with cutting-edge offerings, like Google App Engine, as soon as they're available. We'd especially like to thank Ryan Nichols, V.P. of Cloud Strategy for Appirio, who wrote the fantastic Foreword for this book. Ryan is a thought leader in the cloud computing space and we're honored to have him take an interest in our book.

Finally, thanks to all of you for taking that leap of faith from traditional development environments to cloud-based platform development. App Engine is a key component of cloud computing and will no doubt be a platform that runs some of the most exciting web applications we'll see in the next few years. Hopefully, with this foundation, one (or more) of those applications will be yours!

Kyle Roche
Jeff Douglas

Introduction

Application development, as you know it, is about to change. Think about your development project plans for a moment. Do they all seem to have the same line items for the first phase? Build a server. Install a database. Configure the application server. You're a programmer, aren't you? Why spread your talents so thin? You should be able to focus your energy on building the application from day one. That's where Google App Engine comes into the picture. There's no need to ever worry about building a development server, installing a database, setting up an application server, opening ports, and the endless other tasks that come with traditional development. With Google App Engine, you can start building your application right away.

Google App Engine applications are built using the same tools you use today for Java development. The Google Plugin for Eclipse allows you to develop your entire application in a single IDE. Everything from data management to user-interface design is encompassed in the development environment. You no longer need to use a different tool or server for each layer of the application stack. And most importantly, it's an unquestionable advantage to be able to spend less time on setting up the evironment and more time on the application's business value.

We've been there. We used to spend 80% of our time on application maintenance and upgrades and only 20% on innovation. But the industry is evolving. It's time to reverse that formula. Let Google worry about scalability, security, hosting, load balancing, bandwidth, and all the other preparatory and peripheral tasks that accompany writing an application. We invite you to spend your time innovating and concentrate on the business value of your applications, not their foundations.

In this book we're going to take you through configuring your development environment for Google App Engine. You'll build your first application and quickly advance your way through the offerings that come with App Engine. We'll sprinkle some other technologies into the various chapters—such as Spring, Flex, and Google Web Toolkit (GWT).

This book presents some core examples that build on each other, but for the most part, the chapters are isolated enough to enable you to skip around as needed. In the end you'll build a robust application from the ground up, and there are takeaways from each chapter that you can use in your production environment. And if you are looking for code samples, you've picked up the right book. The book is chock-full of detailed examples of all App Engine's services.

CHAPTER 1

■ ■ ■

Beginning Google App Engine for Java

By now, you've heard about cloud computing. It's gone from a forward-looking concept that was adopted quickly by cutting-edge development communities to a serious requirement for a growing number of businesses. This book focuses on Google App Engine, one of the leading cloud-based development platforms on the market. Powering some of Google's own offerings, like Google Wave and Google Moderator, App Engine provides an affordable, efficient, and scalable platform for developing web applications. App Engine supports both a Java runtime, which we'll cover in this book, and a Python runtime.

Cloud Computing and App Engine

A lot of vendors are staking claims to platform offerings "in the cloud." Currently, it's our opinion that Google, Amazon.com, and Salesforce.com are leading the charge in both the development community and the enterprise-computing space. There are three main, accepted levels of cloud-computing offerings. They are Infrastructure as a Service (IaaS),), Platform as a Service (PaaS),), and Software as a Service (Saas).). Each has unique features and benefits, but there are some commonalities as well.

Any cloud-computing offering should have certain characteristics. Above all, it should be multitenant. A key component of a true cloud-computing platform, multitenancy is a type of software architecture where one instance of the offering is used to serve multiple tenants. The alternative, single tenancy, is how you're probably designing solutions today. Each customer (or business group, or client) gets her own server, database, application layer, and interface. In contrast, a multitenant application would have a single instance of all these layers and would partition the client's data programmatically. Multitenancy is a shared trait among offerings at the IaaS, PaaS, and SaaS layers.

At the lowest level, IaaS offers the physical infrastructure (or virtualized physical infrastructure) to tenants with the ability to pay for what they need in terms of computing power. Instead of purchasing servers, software, and physical location, a tenant of an IaaS offering can pay for these components as needed in a more subscription-based fashion. Leading IaaS vendors like Amazon.com offer “pay per CPU hour” pricing for Linux and Windows platforms. The servers are immediately available and you can spin up dozens of servers in a matter of minutes.

At the highest level, SaaS, much like IaaS, offers solutions to the customer on a per-usage model. The major difference is that SaaS offerings completely abstract the physical and application layers from the end user or developer. For example, Salesforce.com (widely consider the best example of a SaaS offering) provides its own customizable user interface and proprietary programming language (Apex) but doesn’t expose to the end user the hardware or software layers that power the application. SaaS offerings have an important characteristic when it comes to application upgrades and maintenance: everything is centrally updated. So, when a new feature is released or a patch or upgrade is provided, it’s immediately available to all customers.

In between IaaS and SaaS is the PaaS market. PaaS offers a bit more than IaaS, without providing an actual end-user product. PaaS components are typically building blocks or solution stacks that you can use to build your own applications. This is where Google App Engine fits in your cloud-computing portfolio. App Engine is a PaaS offering, currently supporting a Java and a Python runtime to build your scalable web applications without the need for complex underlying hardware and software layers. Google abstracts those layers and lets you concentrate fully on your application. PaaS does have its own set of challenges, however. With PaaS offerings, like App Engine and Force.com, you are restricted by a governor process or application quotas. PaaS governors protect the shared layers of the multitenant platform from being monopolized by one heavy application or runaway code. Application quotas, which Google defines for App Engine applications, define the daily-allotted amount of computing power, space, or bandwidth that any one application is allowed to utilize. With App Engine you have the option to pay for more power or space if needed. See Chapter 2 for more details on the quotas that are defined and their limits.

Consider Figure 1-1 for a moment. Take a look at where the major players sit in relation to the types of cloud offerings we’ve discussed so far as well as in comparison to each other. You can quickly see that the major offerings seem to build on each other. Amazon Web Services, in the bottom-left section, offers the least customization. It simply removes your need to build out a physical infrastructure, leaving all the management and support to your IT staff. Shifting to the right, you see that App Engine offers just slightly more abstraction, now covering the platform and infrastructure. Let’s compare those two scenarios briefly.

Consider a basic J2EE application running on WebSphere. Assume that it meets the requirements for an application that could be run on App Engine. (See Chapter 4 for more information on the restrictions that applications might face on App Engine.) With Amazon's Elastic Computing Cloud (EC2) you can quickly build the Linux stack with a preconfigured Apache server and your choice of Java application server and database. You have to support the operating system, the database, the application server, the security, and all the same components you'd be supporting in an on-premise environment, except the physical machine. This, no doubt, saves time and money. But, IaaS offerings still need provisioning and long-term support at more layers than the application. Now, on the flip side, consider this same application running on App Engine. You don't need hardware provisioned or software installed, and you don't need an application server or a database. All these are wrapped into the core platform offering from Google.

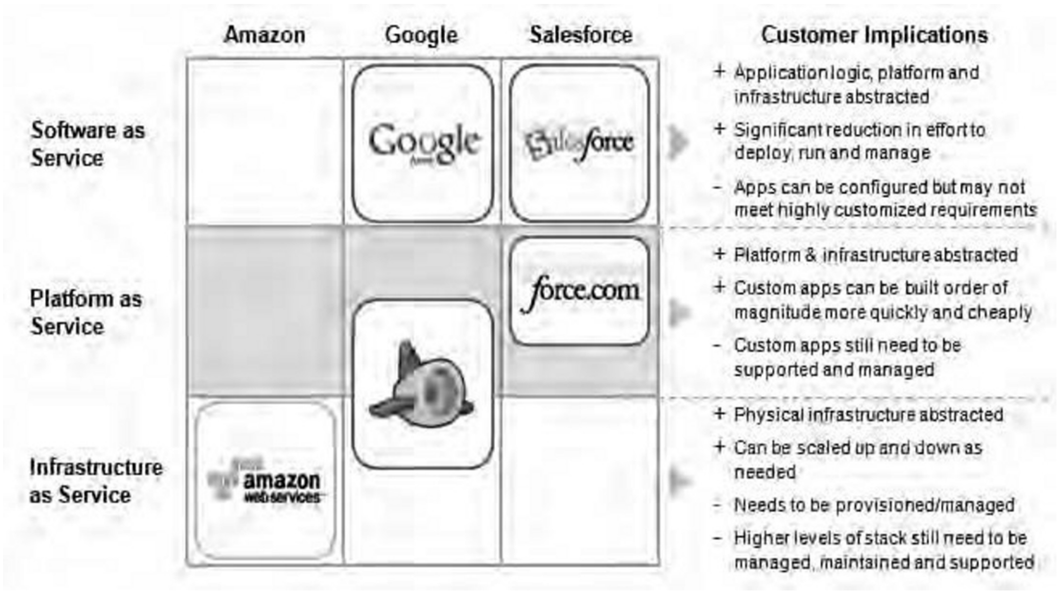
Amazon
Google
Salesforce
Customer Implications
Software as Service
Platform as Service
Infrastructure as Service
+ Application logic, platform and infrastructure abstracted
+ Significant reduction in effort to deploy, run and manage
- Apps can be configured but may not meet highly customized requirements
+ Platform & infrastructure abstracted
+ Custom apps can be built order of magnitude more quickly and cheaply
- Custom apps still need to be supported and managed
+ Physical infrastructure abstracted
+ Can be scaled up and down as needed
- Needs to be provisioned/managed
- Higher levels of stack still need to be managed, maintained and supported

Figure 1-1. *Cloud vendor landscape (Source: Appirio CIO blog)*

Figure 1-1 also shows the Force.com platform in the PaaS sector. It's positioned a bit higher than the App Engine offering, and there's a reason for this. Like some other platform vendors, Force.com encapsulates the runtime environment using its own proprietary language. Apex, the language for Force.com development, looks and feels like Java in many ways but doesn't support the full implementation of any JRE.

It's important to note that the placement of the offerings on this diagram does not indicate preference or correlate with value in any way. Each of these offerings has its own unique value and place in the market. And, in many customer scenarios, we've used a combination of these to build the best solution. In fact, both authors of this book work for a consulting firm (with over 200 people) that has yet to purchase any hardware. We are completely focused on cloud solutions and run our entire business within the three offerings shown in the diagram.

Find More Time to Innovate

Take a look at Figure 1-2, which shows two diagrams comparing the scope of activities and the budget and effort of a traditional IT department with those of another IT department that is leveraging a PaaS offering for its business applications. Take special notice of the amount of maintenance on the hardware, middleware, and application layers for the traditional IT department. It's apparent that all that lost time is intruding on the time, budget, and effort left over for innovation. Now, in comparison, consider the IT department leveraging PaaS offerings for their hardware and middleware layers. Removing the maintenance required to keep those layers in house, the department is free to spend that extra time innovating on its core business applications. You might notice that vendor management is a new time-allotment category when you're using PaaS solutions. However, that's a small effort in comparison to managing these solutions internally.

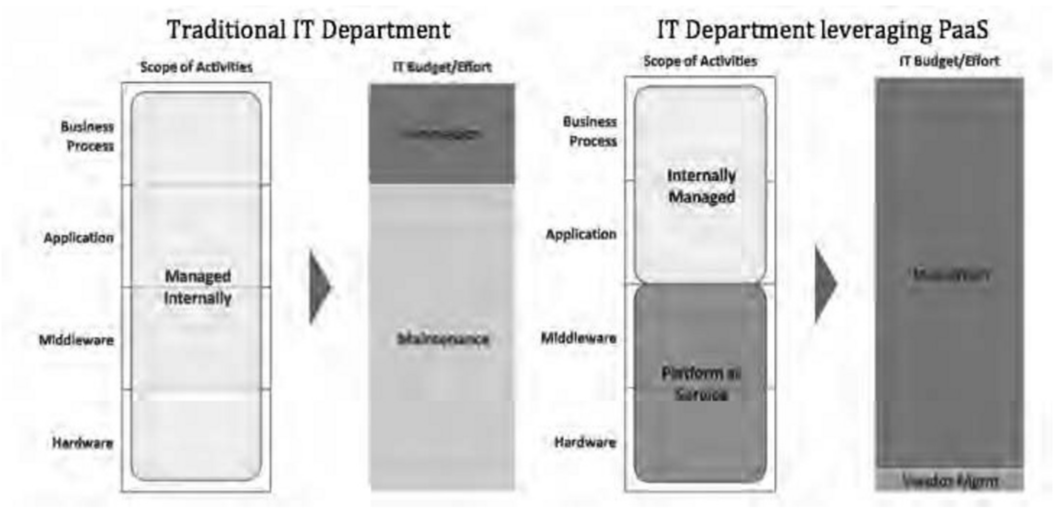

Figure 1-2. *Tradional IT versus IT leveraging PaaS (Source: Appirio CIO blog)*

If you're currently embedded in a traditional software-development structure or a traditional IT department, one of these previous illustrations probably hit home. The inefficiency of traditional IT is one of the main reasons we decided to write this book. The goal was to help you get a jump-start on the major features of Google App Engine for Java, and to give you a platform for building web applications. Let's review some of the skills you're going to learn in the coming chapters.

What You'll Learn in This Book

We've briefly discussed cloud computing and where App Engine fits into the landscape. In Chapter 2 we'll introduce you to more of the underlying architecture for App Engine as well as application quotas. A part of any production application running on App Engine, quotas prevent your application from using too many resources as well as protecting your application from losing resources to other applications.

In Chapter 2, you'll dive right in and sign up for access to App Engine, download the SDK, set up your development IDE, and deploy your first application. If you're going to skip around in the book, make sure you start with Chapter 2, because it lays the foundation and helps you get the tools you'll need to complete the other examples and exercises.

We'll take a step back in Chapters 4 and 5 to tackle a real-world scenario. We'll look at the frameworks and libraries that work well on App Engine and some of the restrictions (and libraries that don't work). Then we'll introduce Google Web Toolkit, and starting from scratch you'll build a timecard application with a rich user interface.

Chapters 6, 7, and 8 cover the service offerings and native tools that come with App Engine. For example, you can leverage Google Authentication services for your applications, which we'll cover in Chapter 6. The App Engine datastore and examples of how to store, query, and index are covered in Chapter 7. In Chapter 8 we'll look at some of the underlying services that the App Engine platform offers your applications. We'll show you how to use App Engine services to send e-mail, send XMPP (Google Talk) messages, manipulate images programmatically, and fetch responses from other web applications.

Finally, we'll cover the Administration Console, the logging functionality, and other maintenance tasks in Chapter 9. We're going to close with a few real-life integration scenarios. First, you'll integrate your App Engine application with Salesforce.com, and then you'll create an App Engine robot for the new and exciting Google Wave offering.

Summary

We have a lot to show you in this book. It's our hope that you'll walk away from it with a solid understanding of the capabilities and features that Google App Engine for Java has to offer. At the time of writing, we covered all the major features of the SDK. If you know Google, you know that they "release early and release often," which makes for a fantastic platform for development as well as a moving target for documentation. Check the online documentation often for updates, and happy coding.

CHAPTER 2

Introduction to App Engine

Google App Engine has been a fantastic disrupter in the technology industry. It's quickly driving innovation among developers and is starting to facilitate a different type of thinking and planning in the enterprise space. App Engine enables you to build enterprise-scalable applications on the same infrastructure that Google uses! The release of Java as the second official language for App Engine marks a tremendous shift in the way applications are being built.

In this chapter we'll cover the basics of App Engine and how it's structured. We'll discuss the major features and benefits of using a platform like App Engine as well as some of the major design considerations (for example, application quotas) that must take place in a multitenant environment.

App Engine Architecture

App Engine is structured differently from the typical web application server. At its core, App Engine restricts your application from any access to the physical infrastructure, preventing you from opening sockets, running background processes (although you can use cron), and using other common back-end routines that application developers take for granted in other environments. Take a look at Figure 2-1. Remember, App Engine is designed to address your concerns about scalability and reliability. It is built on the concept of horizontal scaling, which, in essence, means that instead of running your application on more powerful hardware, you would run your application on more instances of less powerful hardware.

In Figure 2-1 you can see your App Engine application running as an isolated entity within the structure of the multitenant environment. As we discussed in Chapter 1, App Engine shares resources among multiple applications but isolates the data and security between each tenant as well. Your application is able to use some of the Google services, like URL Fetch, to execute processes on its behalf. Because you can't open ports directly within your application, you have to rely on this service, for example, to request Google to open a port and execute the fetch on a URL for the application.

Breaking it down a bit more, consider an apartment building (App Engine) with central air and heating controls. You are a tenant (your App Engine application) in this building. You can't directly adjust the temperature because that would affect the other tenants (other App Engine applications). So, you have to send a request to the building super to change the temperature on your behalf (URLFetch, Bigtable query, Memcache, mail, XMPP, any other Google App Engine service). This is essentially what is happening with App Engine.

If you take a step back, you'll see the long-term implications of this approach. As a developer you now get to ignore scalability concerns like execution time on methods after you have increased data in your datastore. In exchange, you get a fixed duration on execution no matter what your scale becomes. App Engine's response times will be steady from your first request to your millionth request.

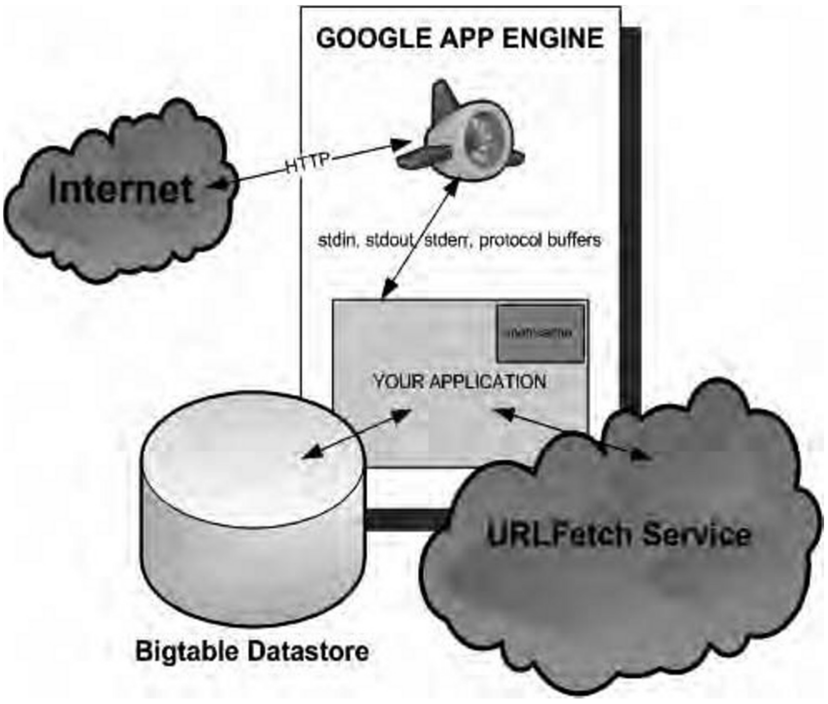

Figure 2-1. App Engine architecture

Notice that no file system or components of the architecture represent the physical machine. With App Engine, you have access only to the application layer. There are some open-source projects, for example, Google Virtual File System, that allow you to