

Adriaan de Jonge


ADDISON-WESLEY

Google App Engine

Tworzenie wydajnych aplikacji w Javie

Od projektu do wdrożenia!



Helion 

Tytuł oryginału: Essential App Engine:
Building High-Performance Java Apps with Google App Engine

Tłumaczenie: Justyna Walkowska

ISBN: 978-83-246-4689-0

Authorized translation from the English language edition, entitled: ESSENTIAL APP ENGINE: BUILDING HIGH-PERFORMANCE JAVA APPS WITH GOOGLE APP ENGINE; ISBN 032174263X; by Adriaan De Jonge; published by Pearson Education, Inc, publishing as Addison Wesley. Copyright © 2012 Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Polish language edition published by HELION S.A. Copyright © 2012.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
<ftp://ftp.helion.pl/przyklady/gooaej.zip>

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<http://helion.pl/user/opinie/gooaej>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Wprowadzenie	13
Podziękowania	21
O autorze	23
I Wprowadzenie do App Engine	25
1 Konfiguracja środowiska	27
Praca z Eclipse	27
Instalacja wtyczek Eclipse	28
Utworzenie nowego projektu App Engine	31
Uruchomienie serwera roboczego	33
Wdrożenie aplikacji na serwerach Google	36
Wdrażanie z linii poleceń	39
Uruchamianie serwera roboczego z linii poleceń	40
Wdrażanie aplikacji App Engine z linii poleceń	40
Podsumowanie	40
2 Wydajność w środowisku App Engine	41
Wydajność w chmurze	41
Porównanie App Engine z tradycyjnymi aplikacjami internetowymi	42
Optymalizacja kosztów zasobów	42
Pomiar kosztów ładowania klas	42
Czas uruchomienia serwletu zawierającego zewnętrzną bibliotekę	43
Czas uruchomienia serwletu niezawierającego zewnętrznej biblioteki	44
Zmniejszenie objętości pliku web.xml	45

Unikanie zimnych uruchomień	47
Rezerwacja stale czynnych instancji	47
Wstępne ładowanie klas przy użyciu żądań rozgrzewających	48
Obsługa współbieżnych żądań w sposób bezpieczny dla wątków	48
Obsługa żądań wymagających dużej ilości pamięci za pomocą instancji typu backend	48
Ogólna poprawa wydajności	49
Optymalizacja modelu danych pod kątem wydajności	49
Unikanie nadmiarowych obliczeń przy użyciu cache	49
Odraczanie długotrwałych zadań za pomocą kolejki zadań	49
Poprawa szybkości ładowania stron w przeglądarce	50
Asynchroniczne API	50
Optymalizacja aplikacji przed jej wdrożeniem	50
Podsumowanie	51

II Podstawy projektowania aplikacji 53

3 Anatomia aplikacji Google App Engine 55

Przesyłanie plików w celu wykonania dynamicznego wdrożenia	55
Struktura katalogów	56
Ustawianie parametrów wdrożenia	58
Uruchamianie zadań okresowych	61
Określenie indeksów w magazynie danych	61
Blokowanie zakresów IP	62
Konfiguracja poziomów logowania	63
Konfiguracja kolejek zadań	63
Zabezpieczanie URL	63
Konfiguracja panelu administracyjnego	64
Podstawy aplikacji	65
Aktualna wersja	65
Dodawanie użytkowników	66
Naliczanie opłat	67
Podsumowanie	67

4 Modelowanie danych na potrzeby magazynu Google App Engine Datastore 69

Odwrot od relacyjnych magazynów danych	69
Denormalizacja danych	70
Agregacja danych bez złączeń	70
Projektowanie danych bezschematowych	71
Modelowanie danych	71
Projektowanie na poziomie mikro	71
Wybór właściwości	73
Rozdzielenie encji	73

Tworzenie i utrzymywanie relacji pomiędzy encjami	74
Relacje jeden do wielu i wiele do wielu	75
Praca z danymi	76
Transakcje	76
Zapytania	77
Tworzenie indeksów	78
Aktualizacja wersji aplikacji korzystającej z Datastore	78
Podsumowanie	79
5 Projektowanie aplikacji	81
Zbieranie wymagań	81
Wybór zestawu narzędzi	82
Wybór frameworku	82
Wybór systemu szablonów	84
Wybór bibliotek	85
Decyzje projektowe	86
Modelowanie danych	86
Modelowanie adresów URL	88
Przechodzenie pomiędzy stronami	89
Podsumowanie	89
III Podstawy projektowania interfejsu użytkownika	91
6 Tworzenie interfejsu użytkownika w HTML5	93
Powitajmy HTML5	93
Stosowanie nowych znaczników HTML5	94
Rysowanie po płótnie	96
Przeciąganie i upuszczanie na stronach	97
Ciekawe elementy formularzy	99
Wykrywanie lokalizacji użytkownika	100
Przechowywanie danych po stronie klienta	101
Przechowywanie danych pomiędzy sesjami	102
Przechowywanie danych sesyjnych	103
Odpytywanie ustrukturyzowanych danych z wykorzystaniem lokalnej bazy danych SQL ...	104
Podsumowanie	106
7 Modyfikacja układu strony za pomocą CSS	107
Wskazywanie elementów za pomocą CSS3	107
Obliczanie szczególowości	108
Identyfikatory	108
Wskazywanie klas	110
Wskazywanie pseudoklas	110
Wskazywanie atrybutów	111

Wskazywanie elementów	112
Wskazywanie pseudoelementów	112
Nowe efekty graficzne CSS3	113
Zaokrąglone krawędzie	115
Animacje 2D	116
Animacje 3D	118
Podsumowanie	119
8 Statyczna interakcja z wykorzystaniem kodu JavaScript	121
Uproszczone przykłady	121
Uporządkowanie kodu HTML dzięki zastosowaniu dyskretnego JavaScriptu	124
Zmniejszenie zależności od JavaScriptu poprzez stopniowe rozszerzanie HTML-a	128
Optymalizacja wydajności za pomocą delegacji zdarzeń	130
Unikanie zmiennych globalnych	132
Podsumowanie	134
9 Dynamiczna interakcja z wykorzystaniem technologii AJAX	135
AJAX na sposób klasyczny, bez frameworków	135
Komunikacja z serwerem za pomocą XML-a	136
Komunikacja z serwerem za pomocą formatu JSON	138
Komunikacja z serwerem za pomocą HTML-a	140
Google App Engine Channel API	141
Otwarcie kanału przez serwer	142
Obsługa komunikatów po stronie klienta	144
Podsumowanie	146
IV Korzystanie z popularnych API App Engine	147
10 Składowanie danych w magazynach Datastore i Blobstore	149
Synchroniczne przetwarzanie danych	149
Synchroniczne składowanie danych	150
Synchroniczne odpytywanie danych	153
Synchroniczne pobieranie danych	154
Asynchroniczne przetwarzanie danych	156
Asynchroniczne składowanie danych	156
Asynchroniczne odpytywanie danych	158
Asynchroniczne pobieranie danych	159
Ustanawianie transakcji	160
Wieloorganizacyjność i przestrzenie nazw	162
Składowanie i pobieranie dużych plików	164
Składowanie dużych plików w Blobstore	164
Odpytywanie Blobstore	166
Pobieranie plików z Blobstore	168

Wysyłanie dużej ilości danych za pomocą Remote API	169
Podsumowanie	170
11 Wysyłanie i odbieranie poczty elektronicznej	173
Wysyłanie rozbudowanych wiadomości e-mail ze znacznikami HTML i załącznikami	173
Parametryzacja ciała wiadomości	176
Zabezpieczenie serwletu	176
Logowanie wysłanych wiadomości na serwerze roboczym	177
Alternatywa — JavaMail API	177
Porównanie API niskopoziomowego z JavaMail	178
Odbieranie poczty	179
Konfiguracja serwletu odbierającego pocztę	179
Zapisywanie odebranej poczty przez serwlet	180
Odbieranie poczty elektronicznej bez JavaMail API	182
Błędy	184
Uwzględnienie wydajności oraz quoty	184
Ile trwa wysłanie wiadomości e-mail?	184
Jaki jest koszt zimnego uruchomienia?	185
Czy odbieranie poczty jest wydajne?	186
Podsumowanie	186
12 Wykonywanie zadań w tle za pomocą Task Queue API oraz cron	187
Kolejkowanie zadań	187
Kolejkowanie wysyłania poczty	188
Konfiguracja kolejek	189
Kontrola limitów	190
Dodatkowe opcje	191
Jak najlepiej wykorzystać kolejki zadań?	194
Planowanie zadań za pomocą cron	195
Konfiguracja zadań za pomocą cron.xml	195
Jak najlepiej wykorzystać zadania cron?	197
Odczyt nagłówków HTTP	197
Podsumowanie	199
13 Przetwarzanie obrazów z wykorzystaniem usługi App Engine Image Service	201
Oszczędne stosowanie Image API	201
Odczyt i zapis obrazów	202
Odczyt danych wprowadzonych przez użytkownika	202
Zapis w magazynie danych	204
Odczyt z magazynu danych	205
Zwrócenie obrazu użytkownikowi	206
Odczyt z pliku	207

Proste przekształcenia	208
Tworzenie miniatur	208
Przycinanie obrazów	210
Obracanie obrazów	211
Przerzucanie obrazów	211
Zaawansowane przekształcenia	211
Podsumowanie	214

14 Optymalizacja wydajności za pomocą pamięci podręcznej cache 215

Podstawowe zastosowanie API memcache	215
Na co uważać, stosując pamięć cache?	215
Pamięć cache i wartości typu String	216
Strategia obsługi cache	218
Redukcja obciążenia App Engine przy użyciu nagłówków ETag	218
Drobnoziarnista pamięć cache	220
Implementacja interfejsu Serializable	221
Umieszczanie w cache obiektów w postaci surowej	222
Zarządzanie cache	222
Unieważnianie elementów w pamięci cache	223
Opróżnianie cache	224
Inne metody przydatne podczas korzystania z cache	225
Wkładanie i wyjmowanie wielu wartości	225
Rejestracja metod obsługi błędów	225
Inkrementacja wartości	225
JSR 107 jako alternatywa dla Memcache Service	226
Podsumowanie	226

15 Pobieranie danych z zewnątrz za pomocą URL Fetch 227

Pobieranie danych z URL za pomocą żądań GET	227
Wykorzystanie standardowego URL Fetch API	228
Wykorzystanie niskopoziomowego URL Fetch API	229
Odczyt wyników	230
Interpretacja wyników	230
Zapis do memcache	230
Zapis do magazynu danych	231
Dodatkowe opcje URL Fetch	231
Kontrola czasu wygaśnięcia	231
Elegancka obsługa wyjątków	233
Wysyłanie danych z formularza	234
Asynchroniczne pobieranie danych	235

Konsumpcja usług sieciowych	237
Usługi REST	237
Komunikacja z SOAP	238
Bezpieczeństwo	238
HTTPS	238
Otwarte porty	238
Podsumowanie	239
16 Zabezpieczanie aplikacji internetowych za pomocą kont Google, OpenID i OAuth	241
Uwierzytelnianie użytkowników w oparciu o konta Google	241
Uwierzytelnianie użytkowników za pomocą OpenID	244
Dostęp z zewnątrz za pomocą OAuth	246
Zabezpieczanie adresów URL w web.xml	248
Wymuszenie uwierzytelnienia	248
Wymuszenie bezpiecznych protokołów	249
Zagadnienia związane z bezpieczeństwem	249
Walidacja danych wejściowych	250
Konfiguracja wieloorganizacyjności	250
Przechowywanie danych osobowych	250
Podsumowanie	251
17 Wysyłanie i odbieranie wiadomości za pomocą XMPP	253
Wysyłanie komunikatów XMPP	253
Odbieranie komunikatów XMPP	255
Odbieranie powiadomień o subskrypcji	257
Odbieranie powiadomień o obecności	260
Podsumowanie	262
V Wdrażanie aplikacji	263
18 Usprawnienie procesu wytwarzania aplikacji	265
Optymalizacja procesu wytwarzania aplikacji internetowych	265
Wejście w skórę kierownika projektu	266
Mniej prac pobocznych	266
Wyznaczenie ostatecznego celu	266
Rezygnacja ze zbędnych czynności	267
Rozszerzanie funkcjonalności	269
Określenie priorytetów	269
Planowanie iteracji	269
Programowanie sterowane eksperymentem	270
Stopniowe wprowadzanie zmian	271

Mierzenie jakości	271
Optymalizacja produktywności programistów	271
Rytuały	272
Nowe języki programowania	272
Zarządzanie czasem i otoczeniem	272
Podsumowanie	273
19 Zapewnienie jakości za pomocą narzędzi pomiarowych	275
Testowanie w środowisku produkcyjnym	275
Racjonalna ocena wartości dodanej testów	276
Testowanie zdroworozsądkowe	276
Minimalizacja kosztów awarii	276
Inne podejście do użyteczności	277
Funkcjonalność przed wyglądem	277
Optymalizacja użyteczności w oparciu o wyniki profilowania i analizy statystyk	278
Sprawdzanie dostępności za pomocą Capabilities API	278
Logowanie nieoczekiwanych zachowań	280
Ciągłe profilowanie w środowisku produkcyjnym	282
Badanie reakcji użytkownika na Twój interfejs	285
Podsumowanie	287
20 Sprzedaż aplikacji	289
Jak podejść do kwestii sprzedaży?	289
Znajomość odbiorcy	290
Dotarcie do odbiorcy	290
Wiadomość w serwisie informacyjnym	290
Pisanie artykułów	291
Blogowanie	291
Twitter	291
Strony na Facebooku	293
Aplikacje na Facebooku	294
Reklamy w Google Apps Marketplace	295
AdWords	296
Optymalizacja aplikacji na potrzeby wyszukiwarek	297
Zakładki społecznościowe	298
Inne sklepy z aplikacjami mobilnymi	298
Zamiana klientów potencjalnych w aktualnych	299
Obsługa procesu płatności	299
Podsumowanie	300
Skorowidz	301

Anatomia aplikacji Google App Engine

W tym rozdziale objaśniam strukturę typowej aplikacji App Engine, przedstawiam sposób organizacji plików przed ich wdrożeniem oraz wskazuję miejsce, w którym pliki są trzymane po wdrożeniu. Początek rozdziału to przegląd serwerów, na których działa aplikacja. W następnej kolejności krótko przedstawiam pliki wchodzące w skład pakietu wdrożeniowego ze szczególnym uwzględnieniem każdego z plików konfiguracyjnych. Rozdział kończy się opisem parametrów konfiguracyjnych ustawianych w panelu administracyjnym.

Przesyłanie plików w celu wykonania dynamicznego wdrożenia

Kiedy wysyłasz aplikację na serwery Google, nie jest ona od razu wdrażana jako działająca instancja aplikacji. Zamiast tego aplikacja jest zapisywana na serwerze, z którego może zostać dynamicznie wdrożona, kiedy zajdzie taka potrzeba.

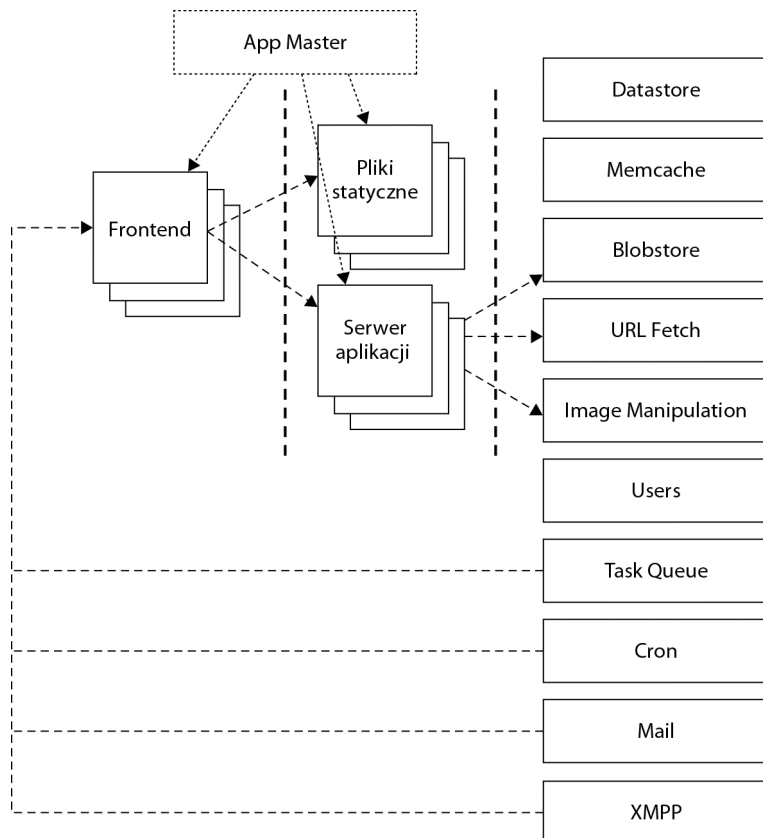
Potrzeba wdrożenia zachodzi wtedy, gdy jeden z serwerów typu *frontend* odbiera żądanie. W tym kontekście frontend to serwer WWW odbierający żądania przed ich przekazaniem do instancji serwera aplikacji.

Na podstawie danych zapisanych w App Masterze frontend podejmuje decyzję, czy żądanie należy przekazać do serwera statycznych plików, czy do jednego z serwerów aplikacji. Serwery uczestniczące w tym procesie zostały przedstawione na rysunku 3.1.

Google skaluje aplikacje, dynamicznie uruchamiając i zatrzymując serwery aplikacji w zależności od aktualnej listy żądań. Aplikacje reagują nie tylko na ilość obciążenia, ale także na jego pochodzenie.

Jeśli na przykład aplikacja zyska popularność w Japonii, Google uruchomi maszyny w (lub niedaleko od) Japonii. Jeśli większość żądań pochodzi z Niemiec, zostaną uruchomione tamtejsze serwery.

Myśląc o wydajności aplikacji App Engine, pamiętaj, że większość usług Google działa na zdalnych serwerach. Wywołanie tych usług zawsze obejmuje pewien narzut komunikacyjny.



Rysunek 3.1. Przegląd serwerów i usług App Engine

Na podstawie rysunku 3.1 łatwo zauważyć, że architektura App Engine została oparta o filozofię, zgodnie z którą *wszystko jest żądaniem HTTP*. Wszystkie żądania do instancji App Engine są przesyłane za pośrednictwem HTTP.

Simple Mail Transfer Protocol (SMTP), Extensible Messaging and Presence Protocol (XMPP), rozkazy Task Queue i zadania cron — wszystkie te elementy są tłumaczone na żądania HTTP przekierowywane przez frontend. Wykorzystanie HTTP ułatwia zarówno obsługę żądań, jak i skalowanie aplikacji App Engine. Skalowanie pod kątem dużej liczby przychodzących wiadomości e-mail przebiega dokładnie tak samo jak skalowanie pod kątem dużej liczby odwiedzin.

Struktura katalogów

Jeśli Twoja aplikacja ma zostać zainstalowana na serwerach App Engine, musisz jej nadać strukturę przypominającą strukturę archiwum WAR, stosowanego podczas wdrażania na takich serwerach aplikacji jak Tomcat czy Jetty. Pamiętaj jednak, że App Engine nie przyjmuje plików WAR, a poza tym archiwum musi zawierać pewne dodatkowe pliki konfiguracyjne.

Zamiast pliku WAR narzędzie wdrożeniowe App Engine oczekuje folderu z rozszerzoną wersją pliku WAR. Po kompilacji typowa aplikacja ma strukturę wyglądającą mniej więcej tak:

```
/moja-aplikacja/  
--/wszelkie-bezpośrednio-dostepne-pliki  
--/WEB-INF  
----/appengine-generated/  
-----/datastore-indexes-auto.xml  
----/classes/ (*)  
-----/najwyzszy-poziom-twoich-pakietow/  
-----/itd./  
----/lib/ (*)  
-----/wymagany-jar-jeden.jar  
-----/wymagany-jar-dwa.jar  
-----/appengine-api-1.5.1.jar  
-----/appengine-api-labs-1.5.1.jar  
----/wszelkie-ukryte-pliki  
----/appengine-web.xml (*)  
----/cron.xml  
----/datastore-indexes.xml  
----/dos.xml  
----/logging.properties  
----/queue.xml  
----/web.xml (*)
```

Pliki oznaczone gwiazdką (*) są niezbędne do działania aplikacji. **Pogrubienie** oznacza, że dany plik w rzeczywistej aplikacji ma dokładnie taką samą nazwę jak na tym listingu. Jeśli nazwa pliku nie została pogrubiona, oznacza to, że została użyta jedynie jako przykład.

Uwaga

W rzeczywistości katalogi `/classes/` i `/lib/` nie są konieczne do uruchomienia aplikacji. Mógłbyś na przykład stworzyć aplikację internetową zawierającą pojedynczy plik JSP niekorzystający z żadnych usług Google. Jednak w większości standardowych aplikacji katalogi te są wymagane.

Mógłbyś także spakować pliki do własnego archiwum JAR i umieścić je w katalogu `/lib/`. Istnienie osobnych katalogów `/classes/` i `/lib/` pozwala jednak na eleganckie rozdzielenie właściwego kodu programu od zewnętrznych bibliotek.

Kiedy porównasz listę bibliotek widoczną w zestawieniu z listą standardowo dołączaną przez narzędzia programistyczne Google, może Ci się ona wydać dosyć krótka. Być może zdążyłeś się już przyzwyczaić do tego, że lista zawiera biblioteki Java Data Objects (JDO), Java Persistence API (JPA), DataNucleus, Java Specification Request (JSR) 107 i być może parę innych. Jeśli jednak nie korzystasz z tych bibliotek, nie musisz dołączać ich do aplikacji. W tej książce nie zachęcam do używania tych dodatkowych bibliotek, dlatego nie będą one dołączane do aplikacji.

Gdy korzystasz z narzędzi programistycznych Google, takich jak plug-in do Eclipse, pozbycie się wspomnianych bibliotek może być nie lada wyzwaniem. Możesz je skasować, ale one i tak co jakiś czas będą do Ciebie wracały. Trwałym rozwiązaniem tego problemu jest rezygnacja z plug-inu i korzystanie z narzędzi konsolowych. Proste narzędzia konsolowe pozwalają na wykonanie każdego zadania realizowalnego z użyciem plug-inu, o ile wiesz, jak skompilować aplikację internetową w języku Java.

Przechodząc na narzędzia konsolowe, uzyskasz większą kontrolę nad tym, jakie pliki będą dodawane do aplikacji. Jeśli będziesz korzystał z narzędzi konsolowych w sposób pokazany w końcowej części rozdziału 1., „Konfiguracja środowiska”, i stworzysz strukturę katalogów analogiczną do tej z ostatniego przykładu, to uzyskasz możliwość wykluczenia z aplikacji każdej biblioteki, której nie potrzebujesz.

Prawdopodobnie zechcesz jednak pozostawić w strukturze katalogów bibliotekę *appengine-api-x.y.z.jar* (gdzie *x.y.z* to numer aktualnej wersji). Bez niej trudno stworzyć jakąkolwiek użyteczną aplikację.

Tworzenie odpowiednich struktur katalogów mogą Ci ułatwić narzędzia automatyzacji budowy, takie jak Maven albo Ant, być może wraz z plug-inem Ivy przeznaczonym dla Ant. Pozwalają one na skompilowanie kodu do żądanej postaci za pomocą polecenia wydawanego z linii komend, niezależnie od tego, jakie środowisko było używane podczas pisania kodu. Budowanie kodu odbywa się wówczas niezależnie od środowiska programistycznego, co przy okazji ułatwia programiście przechodzenie pomiędzy Eclipse, NetBeans i IntelliJ IDEA, kiedy tylko tego zapragnie.

Ustawianie parametrów wdrożenia

Za pomocą pliku *appengine-web.xml* możesz określić parametry konfiguracyjne związane konkretnie z Google App Engine. Większość parametrów definiowanych w tym pliku jest opcjonalna. Jeśli ich nie podasz, App Engine użyje wartości domyślnych.

Identyfikacja aplikacji

Pierwsze dwa elementy, `application` i `version`, są obowiązkowe. Google potrzebuje ich do zidentyfikowania wdrażanej aplikacji. Przykładowe wartości tych parametrów widać na listingu 3.1. Podczas wdrażania zostaniesz poproszony o podanie adresu e-mail oraz hasła, co pozwoli narzędziu wdrożeniowemu na sprawdzenie, czy masz odpowiednie prawa pozwalające na aktualizację danej aplikacji. Kwestia praw zostanie poruszona na końcu tego rozdziału.

Listing 3.1 Przykładowa treść pliku *appengine-web.xml*

```

01 <?xml version="1.0" encoding="utf-8"?>
02 <appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
03   <application>blo-gae</application>
04   <version>1</version>
05
06   <static-files>
07     <include path="/**.jpg" expiration="6h"/>
08     <exclude path="/resources/**/*.xml"/>
09   </static-files>
10
11   <resource-files>
12     <include path="/**/*.xml"/>
13     <exclude path="/static/**/*.jpg"/>
14   </resource-files>
15
16   <!-- Konfiguracja java.util.logging -->
```

```
17 <system-properties>
18   <property name="blog.production-version" value="true"/>
19   <property name="java.util.logging.config.file"
20     value="WEB-INF/logging.properties"/>
21 </system-properties>
22
23 <env-variables>
24   <env-var name="LANGUAGE" value="PL"/>
25 </env-variables>
26
27 <ssl-enabled>false</ssl-enabled>
28
29 <sessions-enabled>true</sessions-enabled>
30
31 <user-permissions>
32   <permission class="com.unknown.CustomPermission"
33     name="custom-name" actions="read"/>
34 </user-permissions>
35
36 <public-root>/my</public-root>
37
38 <inbound-services>
39   <service>mail</service>
40 </inbound-services>
41
42 <precompilation-enabled>true</precompilation-enabled>
43
44</appengine-web-app>
```

Oddzielenie plików statycznych od zasobów

Elementy `static-files` i `resource-files` zajmujące linie od 6. do 14. są związane z wysokopozomową strukturą katalogów, omówioną na początku tego rozdziału. Jeśli nie wyspecyfikujesz tych elementów, wszystkie pliki z wyjątkiem tych z `/WEB-INF/*` zostaną skopiowane zarówno na serwery statycznych plików, jak i na serwery aplikacji. Oznacza to, że zajmą dwa razy więcej miejsca, niż powinny, przez co możesz szybciej przekroczyć wyznaczony limit darmowego miejsca na dysku (quotę).

Katalog `/WEB-INF/` i wszystko poniżej jest traktowane jako zasób. Dla każdego innego pliku powinieneś jawnie określić, czy będzie on odczytywany programistycznie po stronie serwera. Jeśli nie, powinieneś usunąć plik z listy zasobów i dodać go do plików statycznych.

Zasada ta działa także w drugą stronę. Jeśli któryś z plików poza `/WEB-INF/` nigdy nie jest odczytywany bezpośrednio przez przeglądarkę i służy jedynie jako wejście dla programu działającego po stronie serwera, rozważ usunięcie go z listy plików statycznych. Może nawet najlepszym rozwiązaniem byłoby przesunięcie takich plików do folderu `/WEB-INF/`, dzięki czemu zyskasz pewność, że nie zostaną one odczytane nigdzie indziej.

Konfiguracja właściwości systemowych i plików z logami

W pliku `appengine-web.xml` możesz ustawiać właściwości systemowe. Dość ciekawym zastosowaniem tych właściwości jest określenie, gdzie znajduje się plik `logging.properties`. Za jego pomocą możesz określić minimalny poziom błędów, o których komunikaty mają trafić do logów.

Za pomocą zmiennej `env-variables` określasz zmienne środowiskowe. Nie mają one żadnego dodatkowego zastosowania w App Engine. Zmienne środowiskowe można pobrać wewnątrz programu, wywołując metodę `System.getProperty()`.

Sesje

Domyślnie aplikacje Google App Engine są całkowicie bezstanowe i nie wspierają sesji. Jeśli jednak przełączysz element `sessions-enabled` na `true`, będziesz mógł korzystać z klasy `HttpSession` z interfejsu `Servlet`.

Decydując się na wykorzystanie sesji, miej na uwadze, że ich implementacja korzysta zarówno z magazynu `Datastore`, jak i z `memcache`. Ceną jest ilość miejsca na dysku, a potencjalnie także gorsza wydajność związana z odczytem i zapisem danych w `cache`.

Oznacza to, że musisz rozważyć zarówno koszty komunikacji z zewnętrznymi usługami, jak i koszty serializacji obiektów. Wszystkie obiekty dodawane do sesji powinny implementować `java.lang.Serializable`. Garść ostrzeżeń dotyczących stosowania tego interfejsu znajdziesz w rozdziale 14., „Optymalizacja wydajności za pomocą pamięci podręcznej `cache`”.

Bezpieczeństwo aplikacji

W chwili pisania tego tekstu Google App Engine wspiera połączenia `Secure Sockets Layer (SSL)` tylko dla aplikacji działających w domyślnej domenie `appspot`. Jeśli aplikacja jest uruchomiona we własnej domenie użytkownika, wykorzystanie `SSL` nie jest (na razie) możliwe. Trwają prace nad implementacją tego wymagania. Aktualny stan i plany rozwoju App Engine możesz podejrzeć tutaj: <http://code.google.com/intl/pl/appengine/docs/roadmap.html>.

W tej chwili, zakładając, że Twoja aplikacja działa w domenie `appspot`, możesz zażądać zabezpieczenia URL za pomocą `HTTPS`, korzystając z elementu `ssl-enabled`. Decyzja dotyczy wszystkich adresów URL: albo wszystkie są zabezpieczone, albo żaden. Nie ma możliwości wskazania podgrupy, która ma zostać zabezpieczona.

Na potrzeby bibliotek języka Java dostarczających zewnętrzne klasy uprawnień możesz skonfigurować uprawnienia za pomocą elementu `user-permissions`. W przypadku niektórych frameworków jego zastosowanie może być obowiązkowe, jednak w przykładach w tej książce nie będziemy z niego korzystać.

Poza możliwością dodawania i usuwania plików z serwera plików statycznych możesz użyć jeszcze jednego elementu — `public-root` — w celu uniemożliwienia dostępu do plików statycznych. Skorzystanie z tego elementu nie powoduje zmiany lokalizacji plików statycznych ani odpowiadających im adresów URL. Oznacza on tyle, że nie można odwołać się do żadnego katalogu na serwerze plików statycznych poza tym wskazanym jako `public-root`. Jeśli jako `public-root` wskażemy `/static`, nie będziemy mieli dostępu do plików na ścieżce `/inne`. Dostępne będą natomiast pliki zgnieżdżone głębiej we wskazanym katalogu, na przykład `/static/podpoziom/jeszcze-inne`.

Konfiguracja usług

Rysunek 3.1 przedstawia cztery usługi mające dostęp do frontendu w celu wywoływania akcji na serwerach aplikacji: Task Queue, cron, Mail i XMPP. Jedynie Mail i XMPP trzeba jawnie zadeklarować jako włączone w pliku *appengine-web.xml*.

Różnica pomiędzy XMPP i Mail a resztą jest taka, że są one **usługami przyjmującymi**, które odpowiadają na przychodzące z zewnątrz żądania korzystające z protokołów innych niż HTTP. Możesz je włączyć, przypisując wartości `mail` lub `xmpp_message` elementowi `inbound-services`, jak w przykładzie z listingu 3.1 (linia 39.).

Wyłączenie prekompilacji

Z powodów optymalizacyjnych Google App Engine prekompiluje bajtkod przed uruchomieniem aplikacji. W większości przypadków operacja ta nie jest zauważalna. Istnieje jednak co najmniej jeden scenariusz, w którym warto rozważyć jej wyłączenie: jeśli musisz korzystać z podpisanych bibliotek JAR, prekompilacja zepsuje podpis. W takiej sytuacji ustaw wartość elementu `precompilation-enabled` na `false`.

Uruchamianie zadań okresowych

Możesz wykorzystać plik *cron.xml* do planowania wykonania zadań. Usługa cron uruchamia zadania, wysyłając żądania HTTP do serwerów frontend. Przykładowa treść pliku konfiguracyjnego *cron.xml* została przedstawiona na listingu 3.2. Więcej informacji na temat tej usługi i jej konfiguracji znajdziesz w rozdziale 12., „Wykonywanie zadań w tle za pomocą Task Queue API oraz cron”.

Listing 3.2. Przykładowa zawartość pliku *cron.xml*

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <cronentries>
03   <cron>
04     <url>/tasks/cron/mail</url>
05     <description>Wyślij spamerski e-mail...</description>
06     <schedule>every 2 minutes</schedule>
07   </cron>
08   <cron>
09     <url>/tasks/cron/read-rss</url>
10     <description>Odczytaj strumień RSS...</description>
11     <schedule>every day 22:00</schedule>
12     <timezone>Africa/Johannesburg</timezone>
13   </cron>
14 </cronentries>
```

Określenie indeksów w magazynie danych

W strukturze plików na początku rozdziału zauważysz dwa pliki, których nazwy zaczynają się od *datastore-indexes*: są to pliki *datastore-indexes.xml* oraz *datastore-indexes-auto.xml*.

Plik *datastore-indexes.xml* służy do wskazania, które właściwości których encji powinny zostać zindeksowane w celu optymalizacji czasu zapytań do magazynu danych.

Plik *datastore-indexes-auto.xml* w katalogu */WEB-INF/appengine-generated* jest tworzony przez serwer roboczy. Może on współpracować z *datastore-indexes.xml*, o ile wartość atrybutu *autoGenerate* jest ustawiona na *true*. W przeciwnym razie jest używany jedynie plik *datastore-indexes.xml*, a *datastore-indexes-auto.xml* jest ignorowany. Listing 3.3 przedstawia treść pliku *datastore-indexes.xml*, który pozwala serwerowi roboczemu na automatyczne wprowadzanie zmian w pliku *datastore-indexes-auto.xml*.

Listing 3.3. Przykładowa zawartość pliku *datastore-indexes.xml*

```

01 <?xml version="1.0" encoding="utf-8"?>
02 <datastore-indexes
03   autoGenerate="true">
04   <datastore-index kind="BlogPost" ancestor="false">
05     <property name="title" direction="asc" />
06     <property name="entry-date" direction="desc" />
07   </datastore-index>
08
09   <datastore-index kind="User" ancestor="false">
10     <property name="name" direction="asc" />
11   </datastore-index>
12 </datastore-indexes>

```

Więcej informacji na temat tworzenia indeksów i korzystania z nich znajdziesz w rozdziale 4., „Modelowanie danych na potrzeby magazynu Google App Engine Datastore”.

Blokowanie zakresów IP

W celu ochrony aplikacji przed nadużyciami możesz chcieć ograniczyć dostęp do niej za pomocą zakresu adresów IP. Możesz wpisać adres IP lub zakres adresów na „czarną listę” przy użyciu pliku *dos.xml* (nazwa to skrót od *denial of service* — odmowa obsługi).

Listing 3.4 przedstawia przykład zablokowania jednego konkretnego adresu IP z zakresu IPv4 oraz jednej podsieci. Podsieć została zdefiniowana za pomocą notacji Classless Inter-Domain Routing (CIDR). Pełne objaśnienie CIDR wykracza poza zakres tej książki, jednak z łatwością znajdziesz w internecie zasoby opisujące ten koncept.

Listing 3.4. Przykładowa treść pliku *dos.xml*

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <blacklistentries>
03   <blacklist>
04     <subnet>170.224.225.27</subnet>
05     <description>Jeden konkretny adres</description>
06   </blacklist>
07   <blacklist>
08     <subnet>170.224.225.0/24</subnet>
09     <description>Cała podsieć</description>
10   </blacklist>
11 </blacklistentries>

```

Konfiguracja poziomów logowania

Listing 3.1 zawiera odwołanie do pliku *logging.properties*. Za pomocą tego pliku możesz określić poziom logowania, wskazując najmniej ważny poziom komunikatów, które mają już zostać uwzględnione w plikach z logami. Przykład z listingu 3.5 definiuje logowanie wszystkich komunikatów o poziomie co najmniej INFO, niezależnie od pakietu czy klasy.

Listing 3.5. Definicja poziomu logowania w pliku *logging.properties*

```
01 #logging.properties
02 # Ustaw domyślny poziom logowania dla wszystkich loggerów na INFO
03 .level = INFO
```

Inne poziomy to *trace*, *debug*, *warning*, *error* i *fatal*. Możesz wprowadzić różne ustawienia dla różnych części aplikacji, wskazując nazwy pakietów lub pełne nazwy klas przed ciągiem `.level`. Jest to standardowy mechanizm konfiguracji `java.util.logging`.

Konfiguracja kolejek zadań

Jeśli nie stworzysz pliku *queues.xml*, będzie istniała jedna domyślna kolejka zadań o przepustowości pięciu zadań na sekundę. Jeśli chcesz wprowadzić dodatkowe kolejki o różnej przepustowości, możesz dodać plik *queues.xml*, taki jak na listingu 3.6. Więcej informacji na ten temat znajdziesz w rozdziale 12., „Wykonywanie zadań w tle za pomocą Task Queue API oraz cron”.

Listing 3.6. Przykładowa treść pliku *queues.xml*

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <queue-entries>
03   <queue>
04     <name>mail-queue</name>
05     <rate>2000/d</rate>
06   </queue>
07   <queue>
08     <name>second-mail-queue</name>
09     <rate>8/m</rate>
10   </queue>
11 </queue-entries>
```

Zabezpieczanie URL

Większość plików konfiguracyjnych, które się pojawiły w tym rozdziale, była związana ze specyfiką App Engine i nie miałyby sensu przy wdrożeniu aplikacji w innym kontenerze. Jednak istnieje część konfiguracji, która działa tak samo we wszystkich standardowych kontenerach serwletów. Chodzi tu o jedyny pominięty do tej pory plik konfiguracyjny — *web.xml*. Nie wymaga on zbyt obszernych wyjaśnień.

Warto jednak wspomnieć przynajmniej jedną z części tego pliku — ograniczanie dostępu do poszczególnych adresów URL, tak by tylko zalogowani użytkownicy mogli się do nich odwołać. Fragment pliku *web.xml* opisujący adres URL o ograniczonym dostępie przedstawia listing 3.7.

Listing 3.7. Fragment pliku *web.xml*

```

01 <?xml version="1.0" encoding="utf-8"?>
02 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
03     xmlns="http://java.sun.com/xml/ns/javaee"
04     xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
05     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
06     http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
07     version="2.5">
08
09 <!-- [...] -->
10
11 <servlet>
12     <servlet-name>ReceiveMailServlet</servlet-name>
13     <servlet-class>com.appspot.mail.[...]</servlet-class>
14 </servlet>
15 <servlet-mapping>
16     <servlet-name>ReceiveMailServlet</servlet-name>
17     <url-pattern>/_ah/mail/*</url-pattern>
18 </servlet-mapping>
19
20 <!-- [...] -->
21
22 <security-constraint>
23     <web-resource-collection>
24         <web-resource-name>receive-mail-url</web-resource-name>
25         <url-pattern>/_ah/mail/*</url-pattern>
26     </web-resource-collection>
27     <auth-constraint>
28         <role-name>admin</role-name>
29     </auth-constraint>
30 </security-constraint>
31
32 <!-- [...] -->
33
34 </web-app>

```

W linii 28. pojawia się nazwa roli admin. Możliwe są tutaj tylko dwie wartości: * lub admin. Wartość admin oznacza, że tylko zarejestrowani programiści aplikacji lub systemu mogą uzyskać dostęp do danego URL. Z kolei wartość * wskazuje, że dostęp może uzyskać każdy zalogowany użytkownik.

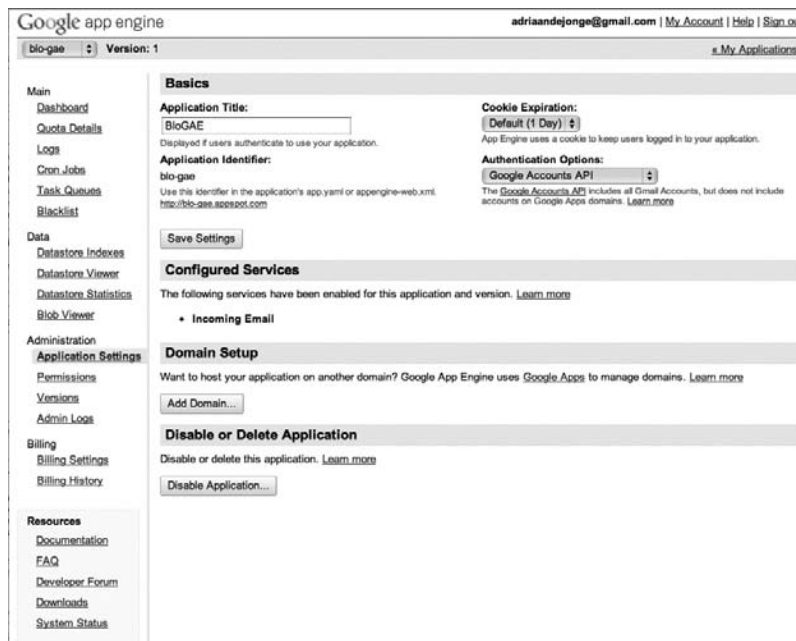
Konfiguracja panelu administracyjnego

Większość ustawień konfiguracyjnych Google App Engine definiuje się w plikach konfiguracyjnych dostarczanych wraz z aplikacją. Jednak pewna niewielka część konfiguracji odbywa się za pośrednictwem panelu administracyjnego. W przypadku większości obecnych tam opcji oczywiście

jest, dlaczego nie trafiły one do plików konfiguracyjnych. Bez wdawania się w filozoficzne dysputy można stwierdzić, że powody są czysto praktyczne. Panel administracyjny jest dość zwięzły i łatwo się w nim odnaleźć.

Podstawy aplikacji

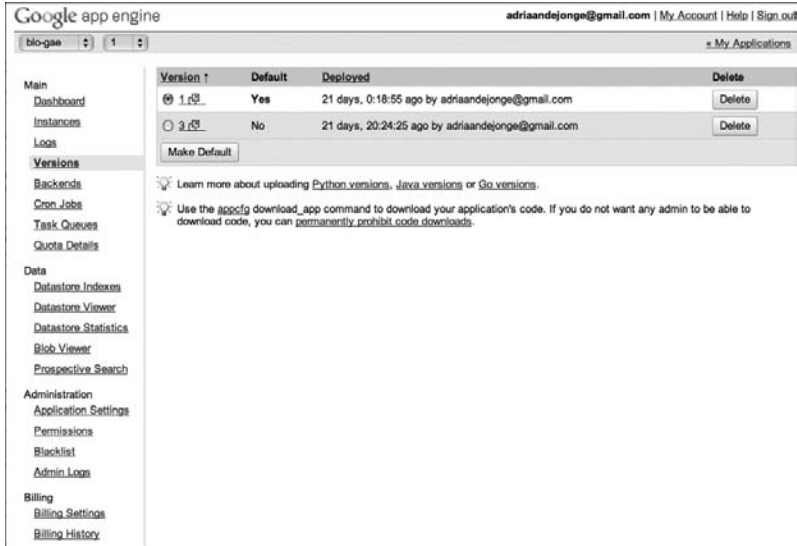
Na ekranie *Application Settings* (ustawienia aplikacji) możesz ustawić kilka podstawowych parametrów, takich jak nazwa aplikacji, czas ważności ciasteczek oraz sposób uwierzytelniania użytkowników. Możesz podać własną nazwę domenową, a także dezaktywować lub skasować aplikację, jeśli nie chcesz już świadczyć usług. Ten ekran panelu administracyjnego został przedstawiony na rysunku 3.2.



Rysunek 3.2. Określenie nazwy aplikacji, dziedziny oraz mechanizmu uwierzytelniania

Aktualna wersja

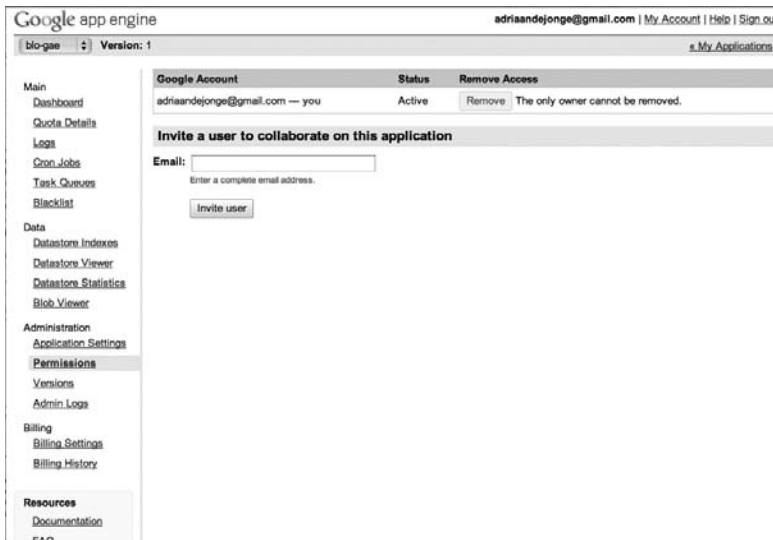
Listing 3.1 zawierał informację na temat wersji. Za pomocą panelu administracyjnego widocznego na rysunku 3.3 możesz przełączyć wersję aplikacji. Umożliwia to przygotowanie się do aktualizacji i podjęcie decyzji, kiedy nowa wersja stanie się aktywna. Możliwe jest także wycofanie wersji w przypadku nieoczekiwanych problemów. Pamiętaj, że niezależnie od wersji aplikacji uruchomiona jest tylko jedna wersja magazynu danych. Dostęp do danych należy implementować, mając na uwadze kompatybilność wstecz i w przód.



Rysunek 3.3. Wybór wersji aplikacji

Dodawanie użytkowników

W miarę zdobywania popularności przez aplikację możesz chcieć dodawać do niej zaufane osoby, które będą sprawowały kontrolę nad sytuacją w czasie, gdy jesteś niedostępny. Możesz nadawać innym osobom prawa w oparciu o konta Google. Tacy użytkownicy mogą zarządzać aplikacjami i dodawać nowe wersje. Ekran dodawania użytkownika został przedstawiony na rysunku 3.4.



Rysunek 3.4. Zapraszanie dodatkowych użytkowników do sprawowania kontroli nad systemem

Naliczanie opłat

Gdy aplikacja stanie się popularna, może się okazać, że wyczerpałeś limit darmowych zasobów Google. Jeśli aplikacji towarzyszy przemyślany model biznesowy, nie powinno to stanowić problemu. Możesz wybrać, jaką ilość i które z dostępnych zasobów chcesz opłacać. Rysunek 3.5 przedstawia ustawienia związane z płatnościami.

The screenshot shows the Google App Engine Billing page for the application 'bio-gae'. The page is titled 'Billing Status: Free' and indicates that the application is operating within the free quota levels. A 'Enable Billing' button is visible. Below this, the 'Billing Administrator' is listed as 'None'. The 'Current Balance' is 'n/a'. The 'Resource Allocations' table is the central focus, showing various resources and their usage relative to the free quota and total daily quota.

Resource	Budget	Unit Cost	Paid Quota	Free Quota	Total Daily Quota
CPU Time	n/a	\$0.10/CPU hour	n/a	6.50	6.50 CPU hours
Bandwidth Out	n/a	\$0.12/GByte	n/a	1.00	1.00 GBytes
Bandwidth In	n/a	\$0.10/GByte	n/a	1.00	1.00 GBytes
Stored Data	n/a	\$0.005/GByte-day	n/a	1.00	1.00 GBytes
Recipients Emailed	n/a	\$0.0001/Email	n/a	2,000.00	2,000.00 Emails
Backend Usage	n/a	Prices	n/a	\$0.72	\$0.72
Always On	n/a	\$0.30/day	n/a	none	
Max Daily Budget:	n/a				

Rysunek 3.5. Konfiguracja płatności za zasoby przekraczające dzienny darmowy limit

Podsumowanie

Po przeczytaniu tego rozdziału powinieneś mieć całkiem niezłe pojęcie na temat konfiguracji aplikacji App Engine. Na początku rozdziału przedstawiłem ogólną architekturę serwerów App Engine i wskazałem w niej miejsce Twojej aplikacji. Okazuje się, że jest ona podzielona pomiędzy serwery aplikacji i serwery statycznych plików. Lokalizacja plików zależy od zdefiniowanej przez Ciebie konfiguracji. Jeśli nie zachowasz ostrożności, pliki statyczne mogą trafić na oba serwery i zająć dwa razy więcej miejsca, niż powinny, przy czym liczba darmowego miejsca na dysku (quota) jest ograniczona. W dalszej części rozdziału przedstawiłem wszystkie istniejące pliki konfiguracyjne App Engine.

Skorowidz

A

AdWords, 296, 297
agregacja danych, 70
AJAX, 135, 136, 140
 frameworki, 135
Always On, instancje, 48
ANTLR, 43, 85
API Blobstore, 165
API, asynchroniczne, 50
aplikacje, 81
 biblioteki, wybór, 85
 decyzje projektowe, 86
 framework, wybór, 82
 mierzenie jakości, 271
 model iteracyjny, 270
 model przyrostowy, 270
 modelowanie danych, 86
 modelowanie URL, 88
 narzędzia, wybór, 82
 odbiorcy, 290
 optymalizacja na potrzeby
 wyszukiwarek, 297
 optymalizacja procesu
 wytwarzania, 265, 266, 267
 rozszerzenie funkcjonalności,
 269
 sprzedaż, 289, 297
 system szablonów, wybór, 84

 wprowadzanie zmian, 271
 zakładki społecznościowe,
 298
 zbieranie wymagań, 81
App Engine, 16
 anatomia aplikacji, 55
 bezpieczeństwo, 238
 cache, 218
 Channel API, 141, 142, 144,
 146
 Datastore, 70, 77, 149, 164
 e-maile, 173
 ETag, nagłówki, 218
 Image API, 201, 202
 optymalizacja kosztów
 zasobów, 42
 panel administracyjny, 64,
 65, 66
 porównanie z tradycyjnymi
 aplikacjami, 42
 przegląd serwerów i usług,
 56
 rejestracja aplikacji, 27
 struktura katalogów aplikacji,
 56, 57, 58
 tryby replikacji danych, 70
 wdrażanie z linii poleceń, 39,
 40

 wdrozenie aplikacji, 36, 37, 38
 wyłączenie prekompilacji, 61
appengine-web.xml, 58, 59, 256,
 257
AppStats, 275, 283
asynchroniczne API, 50
awarie, minimalizacja kosztów,
 276

B

backend, instancje, 49, 199
bąbelkowanie zdarzeń, 131
bezpieczeństwo, 60, 238, 249
 HTTPS, 238
 konfiguracja
 wieloorganizacyjności, 250
 otwarte porty, 238
 przechowywanie danych
 osobowych, 250
 walidacja danych
 wejściowych, 250
Blobstore, 164, 165, 171
 odpytywanie, 166
 pobieranie plików, 168
 składowanie dużych plików,
 164
błędy, obsługa, 184

C

cache, 215, 218, 219, 225
 drobnoziarnisty, 220, 222
 na co uważać, 215, 216
 opróżnianie, 224
 strategia obsługi, 218
 umieszczanie obiektów
 w postaci surowej, 222
 unieważnianie elementów,
 223
 wartości typu String, 216
 zarządzanie, 222
 Capabilities API, 275, 278
 Channel API, 141, 142, 144, 146
 chmura obliczeniowa, 16
 utrzymywanie systemów, 16
 wydajność, 41
 Commons File Upload, 42, 85,
 204
 cron, 187, 194, 195, 200
 planowanie zadań, 195
 wykorzystanie, 197
 cron.xml, 61, 195
 CSS, 17, 119
 atrybuty, 111
 elementy, 112
 identyfikatory, 108
 klasy, 110
 ograniczanie rozmiaru, 108
 pseudoelementy, 112
 pseudoklasy, 110, 111, 113
 wydajność, 108
 CSS3, 107
 animacje 2D, 116
 animacje 3D, 118
 efekty graficzne, 113
 prefiksy, 117, 118
 zaokrąglone krawędzie, 115,
 116
 czas, zarządzanie, 272

D

dane
 agregacja bez złączeń, 70
 bezschematowe, 71

denormalizacja, 70
 modelowanie, 71
 transakcje, 76
 zapytania, 77
 Datastore, 149, 164
 Datastore Viewer, 152
 datastore-indexes.xml, 61
 datastore-indexes-auto.xml, 62
 DeadlineExceededException, 194
 denormalizacja danych, 70
 development server, *Patrz* serwer
 roboczy
 długie sondowanie, 141
 dos.xml, 62
 DTO, 267

E

Eclipse, 27
 instalacja wtyczek, 28
 utworzenie projektu App
 Engine, 31, 32, 33
 ECMAScript 5, 132, 133
 e-maile, 173
 czas wysyłania, 184
 JavaMail, 177
 kontrola limitów, 190
 logowanie wiadomości na
 serwerze roboczym, 177
 odbieranie, 179, 182, 186
 quota, 190
 wydajność, 184
 wysyłanie, 173
 encja, 71, 72
 ETag, 218, 220
 Expires, nagłówek, 219, 220

F

Facebook, 293, 294
 Fluent API, 191
 frameworki
 korzyści, 82, 83
 wady, 83
 FreeMarker, 84
 Future, obiekt, 237

G

Geolocation API, 100
 Google Accounts API, 241, 243
 Google Analytics, 285, 286
 Google Apps Marketplace, 295,
 296
 Google Talk, 257, 260
 Google Website Optimizer, 270

H

high replication, 70
 HTML4, 17
 HTML5, 17, 93, 94, 106
 article, 95
 canvas, 96, 97
 footer, 96
 formularze, 99, 100
 header, 95
 nav, 95
 nowe znaczniki, 94
 przechowywanie danych
 po stronie klienta, 101
 przechowywanie danych
 pomiędzy sesjami, 102
 przechowywanie danych
 sesyjnych, 103
 przeciągnij i upuść, 97, 99
 wykrywanie lokalizacji
 użytkownika, 100
 HTTP Expires, 219
 HTTP, odczyt nagłówków, 197
 HTTPS, 60, 238, 249

I

If-Modified, nagłówek, 220
 If-None-Match, nagłówek, 220
 Image API, 201, 202
 indeksy, 78
 interfejs, badanie reakcji
 użytkowników, 285
 IP, blokowanie zakresów, 62

J

Jabber, 253
 JavaDoc, 268
 JavaMail, 177, 178, 184, 186
 porównanie z API
 niskopoziomowym, 178
 JavaScript, 121, 133
 optymalizacja wydajności, 130
 osadzanie w HTML, 122
 unikanie zmiennych
 lokalnych, 132
 w starszych przeglądarkach,
 126
 wydajność, 123
 język funkcjonalny, 127
 jQuery, 132
 JSON, 138, 140, 146
 JSR 107, 226

K

kierownicy projektu, 266
 kolejki
 konfiguracja, 63, 189
 wykorzystanie, 194
 kolejkovanie zadań, 187
 konfiguracja, 189
 wykorzystanie, 194
 wysyłanie poczty, 188, 190

L

Last-Modified, nagłówek, 220
 Last-Modified-Since, nagłówek,
 220
 logging.properties, 63
 logowanie, konfiguracja
 poziomów, 63
 long polling, *Patrz* długie
 sondowanie

M

mapowanie obiektowo-relacyjne,
 15
 memcache, 215, 216, 230
 czas realizacji wywołania, 216

MemcacheService, interfejs, 225
 modelowanie danych, 71
 model-widok-kontroler,
Patrz MVC
 Moduł Ujawniający,
 wzorzec, 132
 MoSCoW, 269
 multitenancy, *Patrz*
 wieloorganizacyjność
 MVC, 14

N

noscript, 110
 NoSQL, 14, 15

O

OAuth, 246, 247, 248
 obrazy, przetwarzanie, 201
 dodawanie noty o prawach
 autorskich, 211
 miniatury, 208, 210
 obracanie, 211
 odczyt, 202
 odczyt danych
 wprowadzanych
 przez użytkownika, 202
 odczyt z magazynu danych,
 205, 206
 odczyt z pliku, 207
 przekształcenia, 208, 211
 przerzucanie, 211
 przycinanie, 210, 211
 zapis, 202
 zapis w magazynie danych,
 204, 205
 zwrócenie obrazu
 użytkownikowi, 206
 obsługa błędów, 184
 odpytywanie danych
 asynchroniczne, 158
 synchroniczne, 153
 OpenID, 244, 245, 246
 opłaty, 67
 OverQuotaException, 190

P

pamięć podręczna, 215, 216
 panel administracyjny
 konfiguracja, 64, 65, 66
 pliki, składowanie, 164
 pobieranie danych
 asynchroniczne, 159
 synchroniczne, 154
 poczta
 kolejkowanie wysyłania,
 188, 190
 odbieranie, 179, 182, 186
 programiści, optymalizacja
 produktywności, 271
 programowanie defensywne, 216
 programowanie sterowane
 testami, 268
 projektowanie aplikacji, 81
 biblioteki, wybór, 85
 decyzje projektowe, 86
 framework, wybór, 82
 modelowanie danych, 86
 modelowanie URL, 88
 narzędzia, wybór, 82
 system szablonów, wybór, 84
 zbieranie wymagań, 81
 przestrzenie nazw, 162
 przetwarzanie danych
 asynchroniczne, 156
 synchroniczne, 149
 Pull API, 199
 Push API, 199

Q

queue.xml, 63, 190
 quota, 184, 190

R

relacje
 jeden do wielu, 75
 wiele do wielu, 75, 76
 relacyjne bazy danych, 14
 Remote API, 169, 171
 REST API, 199

REST, usługi, 237
 Revealing Module, wzorzec, 132
 roster, 258
 RSS, 197

S

SAX, parser, 230
 Serializable, interfejs, 221
 serializacja, 221, 222
 serwer roboczy

- uruchamianie, 33, 34
- uruchamianie z linii poleceń, 40

 składowanie danych

- asynchroniczne, 156
- synchroniczne, 150

 SOAP, 238
 sprzedaż aplikacji, 289

- AdWords, 296, 297
- blogi, 291
- Facebook, 293, 294
- Google Apps Marketplace, 295, 296
- odbiorcy, 290
- serwisy informacyjne, 290
- Twitter, 291, 292

 SSL, 60
 StringTemplate, 43, 84, 85
 strona, postrzegalny czas ładowania, 124
 subskrypcje, odbieranie powiadomień, 257, 259
 SVG, 96

T

TagSoup, biblioteka, 230
 Task Queue API, 151, 187, 192, 194, 199, 200
 TDD, 268
 testowanie, środowisko produkcyjne, 275, 276
 testy jednostkowe, 267

thread safe, *Patrz* tryb bezpiecznych wątków
 transakcje, 76

- ustanawianie, 160

 trendy internetowe, analiza, 13
 tryb bezpiecznych wątków, 48
 Twitter, 291, 292

U

URL

- modelowanie adresów, 88
- zabezpieczanie, 63
- zabezpieczanie adresów w web.xml, 248

 URL Fetch, 227, 231

- asynchroniczne pobieranie danych, 235
- kontrola czasu wygaśnięcia, 231
- niskopoziomowy, 229
- obsługa wyjątków, 233
- standardowy, 228
- usługi sieciowe, 237
- wysyłanie danych z formularza, 234

 usability, *Patrz* użyteczność
 UserService, 242
 usługi sieciowe, 237
 uwierzytelnianie, 241

- OAuth, 246, 247, 248
- OpenID, 244, 245, 246
- wymuszanie, 248

 użyteczność, 275, 277

- optymalizacja, 278

 użytkownicy

- badanie reakcji na interfejs, 285
- uwierzytelnianie, 241, 244, 245, 246, 248

V

Velocity, 84

W

warmup requests, *Patrz* żądania rozgrzewające
 web.xml, 63, 64, 179

- security-constraint, 176
- zabezpieczanie adresów URL, 248
- zmniejszenie objętości, 45

 wieloorganizacyjność, 162
 wstrzykiwanie zależności, 267
 wydajność, 184

- poprawianie, 49

 wyszukiwanie pełnotekstowe, 87

X

X-AppEngineTaskRetryCount, 199
 XForms, 100
 XHTML 1, 17
 XML, 140, 146
 XMLHttpRequest, 136
 XMPP, 253, 254, 255

- odbieranie komunikatów, 255, 256
- powiadomienia o obecności, 260, 261
- powiadomienia o subskrypcji, 257, 259
- typy komunikatów, 255
- wysyłanie komunikatów, 253, 254

Z

zadania okresowe, 61
 zapytania, 77
 zimne uruchomienia

- koszt, 185
- unikanie, 47

Ż

żądania rozgrzewające, 48

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>



ZAGWARANTUJ
NAJLEPSZĄ WYDAJNOŚĆ
TWOJEJ APLIKACJI
W KAŻDYCH WARUNKACH!

Google App Engine

Tworzenie wydajnych aplikacji w Javie

Google App Engine to idealny przykład usługi PaaS (ang. Platform as a Service). W tym modelu płaci się wyłącznie za wykorzystane zasoby dostawcy. Podważa to za budowanie niezwykle elastycznych rozwiązań informatycznych. Jednak największą zaletą z perspektywy użytkownika tego rozwiązania jest brak konieczności utrzymywania własnej infrastruktury. Niezależnie od sytuacji będziesz zawsze przygotowany na obsłużenie dowolnie dużego ruchu, a to naprawdę się opłaca! Dzięki tej książce błyskawicznie rozpoczniesz przygodę z platformą Google App Engine. Autor pokaże Ci, jak szybko tworzyć złożone i wydajne aplikacje w chmurze Google. Zaprezentuje przemówione techniki pozwalające na skoncentrowanie aplikacji, które są w stanie odpowiedzieć na zapytanie w ciągu dwóch sekund przy tzw. zimnym uruchomieniu i w ciągu co najwyżej setek milisekund podczas normalnego działania w pozostałej części sesji. W trakcie lektury dowiesz się, jak unikać najczęstszych błędów, które dramatycznie pogarszają wydajność i skalowalność aplikacji w chmurze, oraz poznasz najważniejsze technologie do tworzenia interfejsów użytkownika. Proces powstawania aplikacji został omówiony od podziału – od projektowania i modelowania danych, przez bezpieczeństwo i testowanie, aż po wdrożenie.

Po lekturze tej książki:

- swobodnie zainstalujesz aplikację na platformie Google App Engine
- skorzystasz z dodatkowych możliwości platformy
- będziesz przechowywać dane w Datastore'ach
- stworzysz w pełni bezpieczne rozwiązania
- zagwarantujesz Twojej aplikacji najlepszą infrastrukturę!

helion.pl
Księgarnia
Internetowa

№ katalogowy: 0273



Księgarnia Internetowa:
<http://helion.pl>



Zamówienia telefoniczne:
0 801 339900



0 601 339900



Helion

Sprawdź najnowsze promocje

📍 <http://helion.pl/promocje>

📖 Książki i najchętniej czytana

📌 <http://helion.pl/bestsellery>

📞 Zamów informacje o nowościach

📍 <http://helion.pl/nowosc>

Helion SA

ul. Hołdzieczi 3C, 44-100 Głowice

tel. 32 230 90 63

e-mail: helion@helion.pl

<http://helion.pl>



NOO K0RZY 501

ISBN 978-83-246-4689-0



Cena 24,90 zł