Kostenloses eBook

LERNEN

google-app-engine

Free unaffiliated eBook created from
Stack Overflow contributors.

#google-app-engine

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: google-app-engine

It is an unofficial and free google-app-engine ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official google-app-engine.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit der Google-App-Engine

Bemerkungen

Es gibt zwei Möglichkeiten, das GAE SDK (das eigenständige GAE SDK vs. Google Cloud SDK `gcloud`) zu erhalten. Beim Deployment der App mit `gcloud` gibt es leichte Unterschiede. Wenn Sie `gcloud` , können Sie `gcloud app deploy ~/my_app/app.yaml` . Das Verhalten unterscheidet sich von der Verwendung der alten `appcfg.py` . Wenn Sie `appcfg.py` bevorzugen, werden Sie feststellen, dass es nicht verfügbar ist. Dies ist darauf zurückzuführen, dass Google aus bestimmten Gründen beschlossen hat, es vor dem Entwickler zu verbergen. Die `appcfg.py` ist jedoch tatsächlich im Verzeichnis `google-cloud-sdk/platform/google_appengine/` . Weitere nützliche Skripte wie `bulkloader.py` sind ebenfalls vorhanden.

Versionen

Ausführung	Veröffentlichtes Datum
1.9.40	2016-07-15

Examples

Konfiguration

Google AppEngine (GAE) ist ein Platform as a Service (PaaS), mit dem Anwendungen auf "Google Scale" bereitgestellt werden können . Dies ist einer der vielen Dienste der Google Cloud Platform (GCP). Entwickler können andere Dienste wie Google Cloud Storage (GCS) und Google Cloud SQL problemlos in GCP integrieren. Entwickler können einen Satz von Code schreiben, der lokal ausgeführt wird und problemlos auf der Google Cloud Platform bereitgestellt werden kann.

Eine Vogelperspektive für den Einstieg in AppEngine beinhaltet Folgendes:

- Installieren Sie das SDK für Ihre bevorzugte Sprache (Go, Python, Java, PHP, Node.js (in Beta)).
- Verwenden Sie das SDK, um eine Anwendung zu erstellen und lokal zu entwickeln
- Stellen Sie denselben Code, der lokal ausgeführt wird, in einer skalierbaren Laufzeitumgebung bereit

Das AppEngine SDK kann auch mit dem Google Cloud SDK installiert werden:

- Installieren Sie das Google Cloud SDK
- Initialisieren Sie das Google Cloud SDK
- Autorisieren Sie das Google Cloud SDK

- Installieren Sie die GAE-Komponenten. Für Python-Benutzer

  ```
  gcloud components install app-engine-python app-engine-python-extras
  ```

  und für go user,

  ```
  gcloud components install app-engine-go
  ```

  Verwenden `gcloud components list` für andere Sprachen die `gcloud components list` , um die Liste der installierten und verfügbaren Komponenten `gcloud components list` .
- Nach einiger Zeit wird das GAE-SDK installiert.

Andere nützliche Links:

- Formale Google-Dokumentation

Erste Schritte mit der Google-App-Engine online lesen: https://riptutorial.com/de/google-app-engine/topic/971/erste-schritte-mit-der-google-app-engine

Kapitel 2: EdgeCache

Bemerkungen

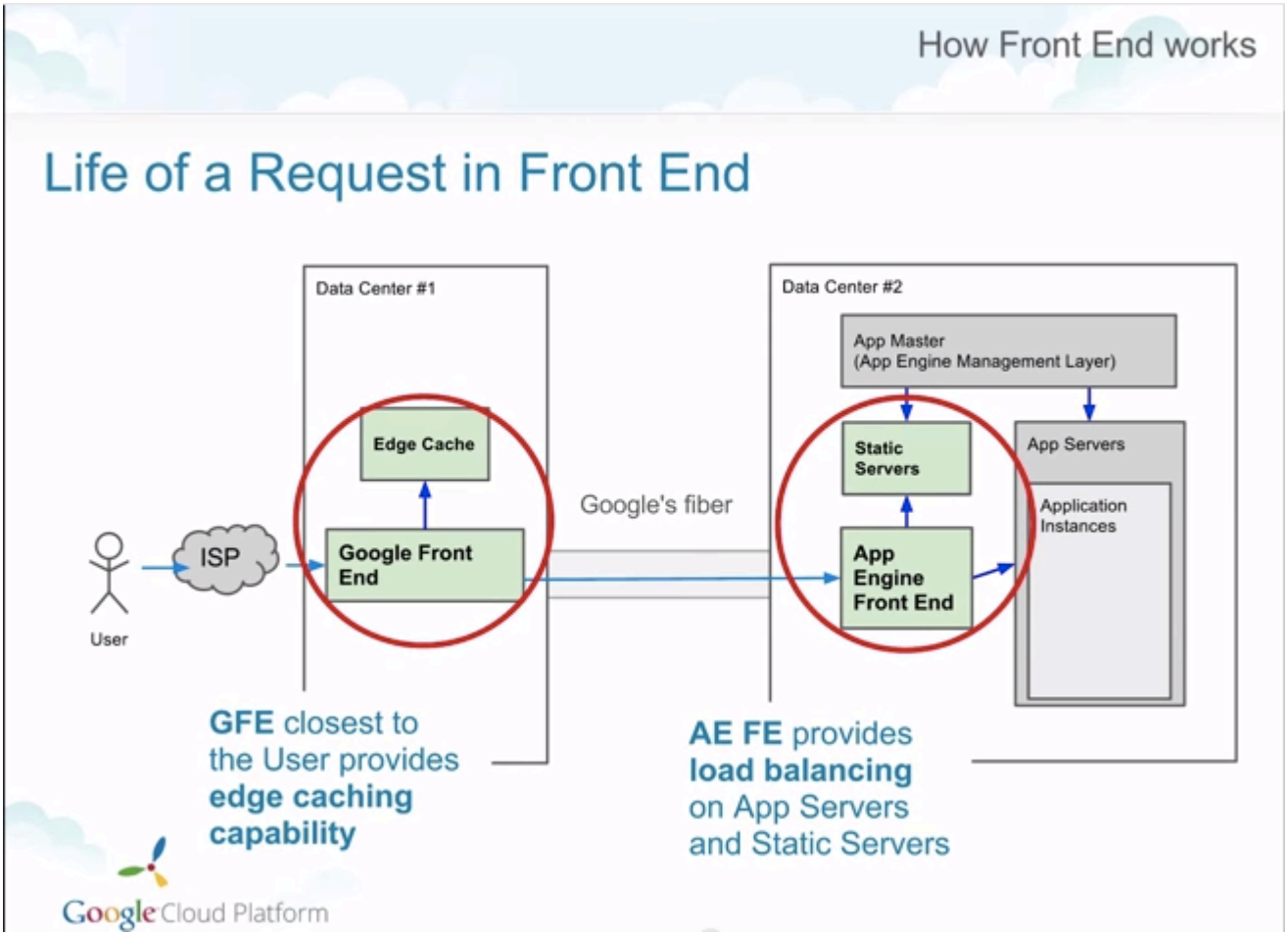
Einzelheiten

- Wenn der Edge-Cache aktiviert ist und funktioniert, sendet App Engine einen `age` dessen Wert die Zeit (in Sekunden) ist, seit die Antwort zwischengespeichert wurde. Wenn beispielsweise die Antwort bisher zwei Minuten zwischengespeichert wurde, enthält die Antwort einen `age: 120` . Wenn kein `age` gesendet wird, ist dies ein gutes Zeichen dafür, dass Sie Edge Cache noch nicht zum Laufen gebracht haben. Wenn die Antwort keinen `age` , bedeutet dies, dass die Anforderung den Edge-Cache verpasst hat.
- Edge Cache funktioniert nicht in der lokalen Entwicklungsumgebung.
- Es gibt verschiedene Caches für verschiedene Regionen. Es gibt separate Caches für einzelne Regionen der Welt. Zum Beispiel könnte eine Anfrage aus Europa den Cache erreichen, während eine Anfrage aus Australien zur selben Zeit nicht den Edge-Cache trifft.
- Es gibt sogar mehrere Caches in derselben Region. Zum Beispiel können zwei aufeinanderfolgende Anforderungen von demselben Client Antworten mit unterschiedlichen Werten für den `age` .
- Google entfernt Inhalte aus Edge Cache, bevor das Höchstalter erreicht ist. Dies gilt insbesondere, wenn die Ressource nicht länger als 5 Minuten angefordert wurde.
- Google sendet automatisch einen `Vary` Header mit dem Wert `Accept-Encoding` und einige Antworten. Siehe diese Dokumentation .
- App Engine lässt es nicht zu, dass der `Cache-Control` Header auf public gesetzt wird, wenn auch ein `Set-Cookie` Header vorhanden ist. App Engine ändert sogar den Wert des `Cache-Control` Headers von `public` in `private` wenn ein `Set-Cookie` Header vorhanden ist. Sehen Sie hier .

Funktionsweise von EdgeCache

EdgeCache ist ein Reverse-Proxy-Cache, in dem Daten für einen bestimmten Zeitraum gespeichert werden und bei einer gleichen Anforderung schnell zurückgegeben werden, solange der Cache noch gültig ist.

Hier ein Diagramm aus dem Video des App Engine-Teams zur Funktionsweise von EdgeCache:



Die "Google Front End" -Datenzentren befinden sich auf der ganzen Welt und können zwischengespeicherte Daten speichern, um sie auf Anfrage schnell zurückzugeben, ohne dass Sie irgendeinen App Engine-Code ausführen müssen (der auf dem "App Engine Front End" ausgeführt wird).

Weitere Informationen zu Reverse-Proxies im Allgemeinen finden Sie in dieser StackOverflow-Antwort .

Momentane Situation

Leider ist der aktuelle Wissensstand über den EdgeCache von GAE ziemlich schlecht. Den Umfang der Dokumentation zu dieser geheimnisvollen Funktion finden Sie in diesem Forumsbeitrag (lesen Sie es jetzt!) Aus dem Jahr 2011 und die 24 Sekunden zwischen 11:11 und 11:35 in diesem Video von Google.

Mehr Ressourcen

Im Folgenden finden Sie eine Liste mit weiteren Ressourcen, die sich auf die EdgeCache-Funktion von App Engine beziehen:

- App Engine-Architektur und Services-Video
- Fantastischer Forumsbeitrag von Brandon Wirtz
- Google App Engine - Problem Nr. 2258 (seit 2009 geöffnet)
- StackOverflow-Frage: Details zum Caching-Proxy von Google App Engine?
- Ein Wort zum Zwischenspeichern der App Engine

- EdgeCache konfigurieren

Examples

EdgeCache aktivieren

- Legen Sie zum Aktivieren von EdgeCache die folgenden HTTP-Antwortheader fest (*Von diesem genauen Format nicht abweichen*):
  - `Cache-Control` Header an `public, max-age=X` wobei:
    - `X = the number of seconds that you want the response to be cached for`
    - `X > 60 seconds`
    - `X < 365*24*60*60`
  - Setzen Sie den `Pragma` Header auf `Public` .

EdgeCache online lesen: https://riptutorial.com/de/google-app-engine/topic/1827/edgecache

Kapitel 3: Google App Engine-Schnellstart für Java

Examples

Bevor Sie beginnen

Bevor Sie dieses Beispiel ausführen, müssen Sie:

Laden Sie das Java SE Development Kit (JDK) herunter und installieren Sie es:

Laden Sie JDK herunter

Laden Sie Apache Maven Version 3.3.9 oder höher herunter:

Installieren und konfigurieren Sie Maven für Ihre lokale Entwicklungsumgebung.

Laden Sie die Hello World App herunter

Wir haben eine einfache Hello World-App für Java entwickelt, mit der Sie schnell ein Gefühl für die Bereitstellung einer App für Google Cloud Platform bekommen können. Führen Sie die folgenden Schritte aus, um Hello World auf Ihren lokalen Computer herunterzuladen.

Klonen Sie das Repository der Hello World-Beispielanwendung auf Ihren lokalen Computer:

```
git clone https://github.com/GoogleCloudPlatform/java-docs-samples.git
```

Wechseln Sie in das Verzeichnis, das den Beispielcode enthält:

```
cd java-docs-samples/appengine/helloworld
```

In den resultierenden `helloworld` Dateien finden Sie das `src` Verzeichnis für ein Paket mit dem Namen `com.example.appengine.helloworld` , das ein einfaches `HTTPServlet` .

Alternativ können Sie das Beispiel als ZIP-Datei herunterladen und extrahieren.

Testen Sie die Anwendung

Testen Sie die Anwendung mithilfe des Entwicklungs-Webservers, der im App Engine SDK enthalten ist.

1. Führen Sie den folgenden Maven-Befehl in Ihrem `helloworld` Verzeichnis aus, um Ihre App zu kompilieren und den Entwicklungs-Webserver zu starten:

   mvn appengine: devserver

Der Webserver wartet jetzt auf Anforderungen an Port 8080.

2. Besuchen Sie http: // localhost: 8080 / in Ihrem Webbrowser, um die App in Aktion zu sehen.

Weitere Informationen zum Ausführen des Entwicklungs-Webservers finden Sie in der Referenz zu Java Development Server .

Nehmen Sie eine Änderung vor

Sie können den Webserver während der Entwicklung Ihrer Anwendung laufen lassen. Wenn Sie eine Änderung vornehmen, erstellen und aktualisieren Sie Ihre App mit dem Befehl `mvn clean package` .

1. Probieren Sie es jetzt aus: Lassen Sie den Webserver laufen und bearbeiten Sie `HelloServlet.java` , um `Hello, world` zu etwas anderem zu ändern.
2. Führen Sie das `mvn clean package` , und laden Sie anschließend http: // localhost: 8080 / neu , um die Ergebnisse anzuzeigen .

Stellen Sie Ihre App bereit

Um Ihre App für App Engine bereitzustellen, müssen Sie ein Projekt registrieren, um Ihre Projekt-ID zu erstellen, die die URL für die App bestimmt.

1. Wechseln Sie in der Cloud Platform Console zur Seite Projekte, und wählen Sie ein neues Projekt aus oder erstellen Sie ein neues.

2. Notieren Sie sich die Projekt-ID, die Sie oben erstellt haben, und geben Sie sie in src / main / webapp / WEB-INF / appengine-web.xml ein. In dieser Datei können Sie auch die App-Version festlegen.

3. Laden Sie Ihre Anwendung zu Google App Engine hoch, indem Sie den folgenden Befehl aufrufen.

   Mvn Appengine: Update

4. Ihre App ist jetzt bereit und bereit, den Datenverkehr unter http: // <YOUR_PROJECT_ID> .appspot.com / bereitzustellen.

Google App Engine-Schnellstart für Java online lesen: https://riptutorial.com/de/google-app-engine/topic/6831/google-app-engine-schnellstart-fur-java

Kapitel 4: Komponententest mit Datastore

Examples

Erstellen Sie einen Kontext mit einem stark konsistenten Datenspeicher.

Beim Testen mit der Testbibliothek von Google App Engine sind die Herausforderungen einer eventuellen Konsistenz auf dieselbe Weise vorhanden, wie sie in der Produktion sind. Um also etwas in den Datastore zu schreiben, um das Abrufen zu testen, müssen Sie einen Kontext erstellen, der stark konsistent ist.

```
type Foo struct {
    Bar string
}

func TestDataStore(t *testing.T) {
    inst, err := aetest.NewInstance(
        &aetest.Options{StronglyConsistentDatastore: true})
    if err != nil {
        t.Fatal(err)
    }
    defer inst.Close()

    req, err := inst.NewRequest("GET", "/", nil)
    if err != nil {
        t.Fatal(err)
    }

    ctx := appengine.NewContext(req)

    foo := &Foo{ Bar: "baz" }

    key, err := key := datastore.NewIncompleteKey(context, "Foo", nil)
    if _, err := datastore.Put(context, key, details); err != nil {
        t.Fatalf(err)
    }

    query := datastore.NewQuery("Foo").Filter("Bar =", "baz")
    for iterator := query.Run(ctx); ; {
        item := &Foo{}
        err := iterator.Next(item)
        if err == datastore.Done {
            t.Fatalf("No results")
        }
        if err != nil {
            t.Fatal(err)
        }
        if foo.Bar != item.Bar {
            t.Fatal("Wrong result returned.")
        }
    }
}
```

Komponententest mit Datastore online lesen: https://riptutorial.com/de/google-app-

engine/topic/6587/komponententest-mit-datastore

Kapitel 5: Python-Laufzeitbeispiele für Google Appengine

Examples

NDB mit Python in AppEngine

In NDB werden Modelle als Python-Objekte bezeichnet, die im Appengine NoSQL-Datastore gespeichert und abgerufen werden können und für alle AppEngine-Anwendungen verfügbar sind.

models.py

```
from google.appengine.ext import ndb

# https://cloud.google.com/appengine/docs/python/ndb/properties


class Series(ndb.Model):
    """TV Series Object"""
    folder_name = ndb.StringProperty()
    title = ndb.StringProperty()
    rating = ndb.StringProperty()
    banner_blob_key = ndb.BlobKeyProperty()
    year = ndb.IntegerProperty()
    plot = ndb.TextProperty()
    genre = ndb.StringProperty(repeated=True)
    json_of_show = ndb.JsonProperty()
    date_added = ndb.DateTimeProperty(auto_now_add=True)
    date_updated = ndb.DateTimeProperty(auto_now=True)


class Episode(ndb.Model):
    """Episode Object (Series have Episodes)"""
    series = ndb.KeyProperty(kind=Series)
    episode_title = ndb.StringProperty()
    season = ndb.IntegerProperty()
    episode_number = ndb.IntegerProperty()
    thumb_blob_key = ndb.BlobKeyProperty()
    episode_json = ndb.JsonProperty()
    date_added = ndb.DateTimeProperty(auto_now_add=True)
    date_updated = ndb.DateTimeProperty(auto_now=True)
```

Wenn unsere Modelle definiert sind, können wir neue Objekte für den Eintrag in den Datastore erstellen:

```
nfo = xmltodict.parse(my_great_file.xml)
s = Series()
s.folder_name = gcs_file.filename[:-10]
s.title = nfo['tvshow'].get('title', None)
s.rating = nfo['tvshow'].get('rating', None)
#  Below we use the google cloud storage library to generate a blobkey for a GCS file
s.banner_blob_key = BlobKey((blobstore.create_gs_key('/gs' + gcs_file.filename[:-10] +
```

```
'banner.jpg')))
s.year = int(nfo['tvshow'].get('year', None))
s.plot = nfo['tvshow'].get('plot', None)
#  genre is a repeated type, and can be stored as a list
s.genre = nfo['tvshow'].get('genre', 'None').split('/')
s.json = json.dumps(nfo)
s.put_async()  #put_async writes to the DB without waiting for confirmation of write.
```

Eine Episode hinzufügen und einer Serie zuordnen:

```
nfo = xmltodict.parse(my_great_file.xml)

epi = Episode()
epi.show_title = nfo['episodedetails'].get('showtitle', None)
epi.title = nfo['episodedetails'].get('title', None)

#  We'll query the Series for use later
show_future = Series.query(Series.title == epi.show_title).get_async()

epi.json = json.dumps(nfo)
... #  We perform other assorted operations to store data in episode properties


#  Ask for the show we async queried earlier
show = show_future.get_result()
#  Associate this episode object with a Series by Key
epi.series = show.key
epi.put_async()  #  Write the object without waiting
```

Später, um alle Serien abzurufen:

```
shows = Series.query()
```

Filter könnten angewendet werden, wenn nicht alle Shows gewünscht wurden.

Mehr lesen:

- Leben eines Datastores Schreiben

Python-Laufzeitbeispiele für Google Appengine online lesen: https://riptutorial.com/de/google-app-engine/topic/5902/python-laufzeitbeispiele-fur-google-appengine

Kapitel 6: Schneller Einstieg mit der Benutzer-Python-API und der App Engine-Authentifizierung

Einführung

Die Verwendung der Benutzer-API ist eine sehr einfache und flexible Möglichkeit, die Authentifizierung in App Engine durchzuführen. Stellen Sie jedoch sicher, dass Ihre Anwendungsfälle keine weiteren Elemente für die Authentifizierungsumgebung erfordern.

Hinweis: Wenn Sie weitere Informationen zur traditionellen Struktur einer App Engine-App benötigen, lesen Sie diese Informationen .

Bemerkungen

Die Benutzer-API ermöglicht:

- Ermitteln Sie, ob sich der aktuelle Benutzer angemeldet hat.
- Leiten Sie den Benutzer zur entsprechenden Anmeldeseite um, um sich anzumelden.
- Bitten Sie Ihren Anwendungsbenutzer, ein neues Google-Konto zu erstellen, falls noch kein Konto vorhanden ist

Referenz und weitere Details

Wichtige Elemente in der Ansicht:

Einführen:

```
from google.appengine.api import users
```

Benutzerobjekt und Methoden:

```
user = users.get_current_user()
```

Hinweis: Die Implementierung von jinja2 ist optional. In diesem Artikel wird jedoch der vollständige Arbeitsablauf erläutert.

Examples

MainPage-Handler [views.py]

Allgemeine Importe: Verwenden Sie jinja2, um Vorlagen in HTML-Dateien zu füllen.

```
import jinja2
import webapp2
```

Wichtiger Import für die Benutzer-API:

```
from google.appengine.api import users
```

Einstellung der Jinja-Umgebung: [im Beispiel die ausgewählte Technologie, um die Informationen in das Frontend einzugeben]

```
JINJA_ENVIRONMENT = jinja2.Environment(
       loader=jinja2.FileSystemLoader(os.path.dirname(__file__)),
       extensions=['jinja2.ext.autoescape'],
       autoescape=True)
```

Konkreter Handler:

```
class MainPage(webapp2.RequestHandler):
   def get(self):

       user = users.get_current_user()
       if user:
           url = users.create_logout_url(self.request.uri)
```

Sie können hier mehr Logik für *Benutzer hinzufügen*

```
       else:
           url = users.create_login_url(self.request.uri)
```

Vorlagen zum Übergeben von Informationen mit jinja2. In diesem Beispiel das Benutzerobjekt und die URL-Zeichenfolge.

```
       template_values = {
           'user': user,
           'url': url,
       }

       JINJA_ENVIRONMENT.add_extension('jinja2.ext.do')
```

Beispiel für index.html. [traditionelle HTML-Seite]

```
       template = JINJA_ENVIRONMENT.get_template('index.html')
       self.response.write(template.render(template_values))
```

App-Routing [urls.py]

Ich habe für dieses Beispiel webapp2 verwendet, um das Routing abzudecken.

```
from webapp2_extras.routes import RedirectRoute as Route
```

Importieren aus Ansichten:

```
from views import MainPage
```

MainPage ist der in root "/" festgelegte Handler:

```
urlpatterns = [
   Route('/', MainPage),
]
```

HTML, Frontend-Beispiel für die Verwendung der Benutzer-API [index.html]

Einfacher Auszug aus index.html:

```
    <div class="sign-in">

        {% if user %}
```

[Übergeben der URL haben wir die Möglichkeit, den Benutzer abzumelden]

```
            <a href="{{ url|safe }}">LOG OUT</a>
```

[Sie können hier Vorgänge für authentifizierte Benutzer einschließen.]

```
        {% else %}
```

[Übergeben der URL haben wir die Möglichkeit, den Benutzer anzumelden]

```
            <a href="{{ url|safe }}">SIGN IN</a>

        {% endif %}
    </div>
```

Dies ist ein einfaches Beispiel für die Verwendung der Operation auf der Seite index.html.

Schneller Einstieg mit der Benutzer-Python-API und der App Engine-Authentifizierung online lesen: https://riptutorial.com/de/google-app-engine/topic/10002/schneller-einstieg-mit-der-benutzer-python-api-und-der-app-engine-authentifizierung

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit der Google-App-Engine	4444, Ari Molzer, Chandana, Community, Dan O'Boyle, Edward Fung, Ezequiel Jadib, Harshal Patil, Paritosh Walvekar
2	EdgeCache	Ani, michaelrbock
3	Google App Engine-Schnellstart für Java	Chandana
4	Komponententest mit Datastore	MrWizard54
5	Python-Laufzeitbeispiele für Google Appengine	Dan O'Boyle
6	Schneller Einstieg mit der Benutzer-Python-API und der App Engine-Authentifizierung	Nicolas Bortolotti