eBook Gratuit

APPRENEZ google-app-engine

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#google-app-engine

Table des matières

À propos

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

Chapitre 1: Démarrer avec google-app-engine

Remarques

Il existe deux manières d'obtenir le SDK GAE (le SDK GAE vs Google Cloud SDK `gcloud`). Il existe de légères différences lors du déploiement de l'application à l'aide de `gcloud` . Si vous utilisez `gcloud` , vous pouvez utiliser l' `gcloud app deploy ~/my_app/app.yaml` . Le comportement est différent de l'utilisation de l'ancien `appcfg.py` . Si vous préférez utiliser `appcfg.py` , vous constaterez qu'il n'est pas disponible. C'est parce que pour certaines raisons, Google a décidé de le cacher au développeur. Cependant, `appcfg.py` est en fait installé dans le répertoire `google-cloud-sdk/platform/google_appengine/` . D'autres scripts utiles tels que `bulkloader.py` sont également disponibles.

Versions

Version	Date de parution
1.9.40	2016-07-15

Examples

Installer

Google AppEngine (GAE) est une plate-forme en tant que service (PaaS) qui permet de déployer des applications sur «Google Scale». C'est l'un des nombreux services de Google Cloud Platform (GCP). Les développeurs peuvent facilement intégrer d'autres services tels que Google Cloud Storage (GCS) et Google Cloud SQL sur GCP. Les développeurs peuvent écrire un ensemble de codes qui s'exécutent localement et peuvent être facilement déployés sur Google Cloud Platform .

Une vue d'ensemble de la prise en main d'AppEngine comprend les éléments suivants:

- Installez le SDK pour votre langue préférée (Go, Python, Java, PHP, Node.js (en version bêta))
- Utiliser le SDK pour échafauder une application et développer localement
- Déployer le même code qui s'exécute localement sur un environnement d'exécution évolutif

Le SDK AppEngine peut également être installé à l'aide du SDK Google Cloud:

- Installer le SDK Google Cloud
- Initialiser le SDK Google Cloud
- Autoriser le SDK Google Cloud
- Installez les composants GAE. Pour l'utilisateur python,

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

et pour aller utilisateur,

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

Pour les autres langages, utilisez la `gcloud components list` pour obtenir la liste des composants installés et disponibles.

- Après un certain temps, le SDK GAE sera installé.

Autres liens utiles:

- Documentation Google officielle

Lire Démarrer avec google-app-engine en ligne: https://riptutorial.com/fr/google-app-engine/topic/971/demarrer-avec-google-app-engine

Chapitre 2: Démarrage rapide avec les utilisateurs API Python, authentification App Engine

Introduction

L' utilisation de l'API Users est un moyen très simple et flexible de travailler avec l'authentification dans App Engine, mais assurez-vous que vos applications ne nécessitent pas plus d'éléments pour l'environnement d'authentification.

Remarque: Si vous avez besoin de plus d'informations sur la structure traditionnelle d'une application App Engine, consultez ces informations .

Remarques

L'API Utilisateurs permet:

- Détecter si l'utilisateur actuel s'est connecté
- Rediriger l'utilisateur vers la page de connexion appropriée pour vous connecter.
- Demandez à votre utilisateur d'application de créer un nouveau compte Google s'il n'en possède pas déjà un.

Référence et plus de détails

Éléments importants dans la vue:

Importer:

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

Objet utilisateur et méthodes:

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

Remarque: l'implémentation de jinja2 est facultative, mais dans l'article est utilisée pour expliquer le flux de travail complet.

Examples

MainPage Handler [vues.py]

Importations générales, en utilisant jinja2 pour remplir les modèles en HTML.

```
import jinja2
import webapp2
```

Import important à utiliser API utilisateurs:

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

Réglage de l'environnement Jinja: [dans l'exemple de la technologie sélectionnée pour remplir les informations dans le frontend]

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

Manutentionnaire de béton:

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

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

Vous pouvez inclure plus de logique ici pour les *utilisateurs*

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

Modèles pour transmettre des informations à l'aide de jinja2. Pour cet exemple, l'objet utilisateur et la chaîne d'URL.

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

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

En utilisant l'exemple index.html. [page HTML traditionnelle]

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

Routage des applications [urls.py]

J'ai utilisé pour cet exemple webapp2 pour couvrir le routage.

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

Importer des vues:

```
from views import MainPage
```

MainPage est le gestionnaire défini dans root "/":

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

Html, exemple frontal d'utilisation de l'API Utilisateurs [index.html]

Extrait simple de index.html:

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

        {% if user %}
```

[En passant l'URL, nous avons la possibilité de déconnecter l'utilisateur]

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

[Vous pouvez inclure ici des opérations pour l'utilisateur authentifié]

```
        {% else %}
```

[En passant l'URL, nous avons la possibilité de connecter l'utilisateur]

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

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

Voici un exemple simple d'utilisation de l'opération sur la page index.html.

Lire Démarrage rapide avec les utilisateurs API Python, authentification App Engine en ligne: https://riptutorial.com/fr/google-app-engine/topic/10002/demarrage-rapide-avec-les-utilisateurs-api-python--authentification-app-engine

Chapitre 3: Démarrage rapide de Google App Engine pour Java

Examples

Avant que tu commences

Avant d'exécuter cet exemple, vous devez:

Téléchargez et installez le kit de développement Java SE (JDK):

Téléchargez JDK

Téléchargez Apache Maven version 3.3.9 ou supérieure:

Installez et configurez Maven pour votre environnement de développement local.

Téléchargez l'application Hello World

Nous avons créé une simple application Hello World pour Java afin que vous puissiez rapidement vous familiariser avec le déploiement d'une application sur Google Cloud Platform. Suivez ces étapes pour télécharger Hello World sur votre ordinateur local.

Clonez le référentiel d'applications Hello World sur votre ordinateur local:

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

Accédez au répertoire contenant le code exemple:

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

Dans les fichiers `helloworld` résultants, vous trouverez le répertoire `src` pour un paquet appelé `com.example.appengine.helloworld` qui implémente un `HTTPServlet` simple.

Vous pouvez également télécharger l'exemple sous forme de fichier zip et l'extraire.

Tester l'application

Testez l'application à l'aide du serveur Web de développement fourni avec le SDK App Engine.

1. Exécutez la commande Maven suivante depuis votre répertoire `helloworld` pour compiler votre application et démarrer le serveur Web de développement:

   mvn appengine: devserver

Le serveur Web écoute maintenant les demandes sur le port 8080.

2. Visitez http: // localhost: 8080 / dans votre navigateur Web pour voir l'application en action.

Pour plus d'informations sur l'exécution du serveur Web de développement, consultez la référence Java Development Server .

Faire un changement

Vous pouvez laisser le serveur Web en marche pendant que vous développez votre application. Lorsque vous apportez une modification, utilisez la commande `mvn clean package` pour créer et mettre à jour votre application.

1. Essayez-le maintenant: laissez le serveur Web en cours d'exécution, puis modifiez `HelloServlet.java` pour changer `Hello, world` en autre chose.
2. Exécutez `mvn clean package` , puis rechargez http: // localhost: 8080 / pour voir les résultats.

Déployez votre application

Pour déployer votre application sur App Engine, vous devez enregistrer un projet pour créer votre ID de projet, qui déterminera l'URL de l'application.

1. Dans la console Cloud Platform, accédez à la page Projets et sélectionnez ou créez un nouveau projet.

2. Notez l'ID de projet que vous avez créé ci-dessus et entrez-le dans src / main / webapp / WEB-INF / appengine-web.xml. Vous pouvez également définir la version de l'application dans ce fichier.

3. Téléchargez votre application sur Google App Engine en appelant la commande suivante.

   mvn appengine: mise à jour

4. Votre application est maintenant déployée et prête à servir le trafic sur http: // <YOUR_PROJECT_ID> .appspot.com /.

Lire Démarrage rapide de Google App Engine pour Java en ligne: https://riptutorial.com/fr/google-app-engine/topic/6831/demarrage-rapide-de-google-app-engine-pour-java

Chapitre 4: EdgeCache

Remarques

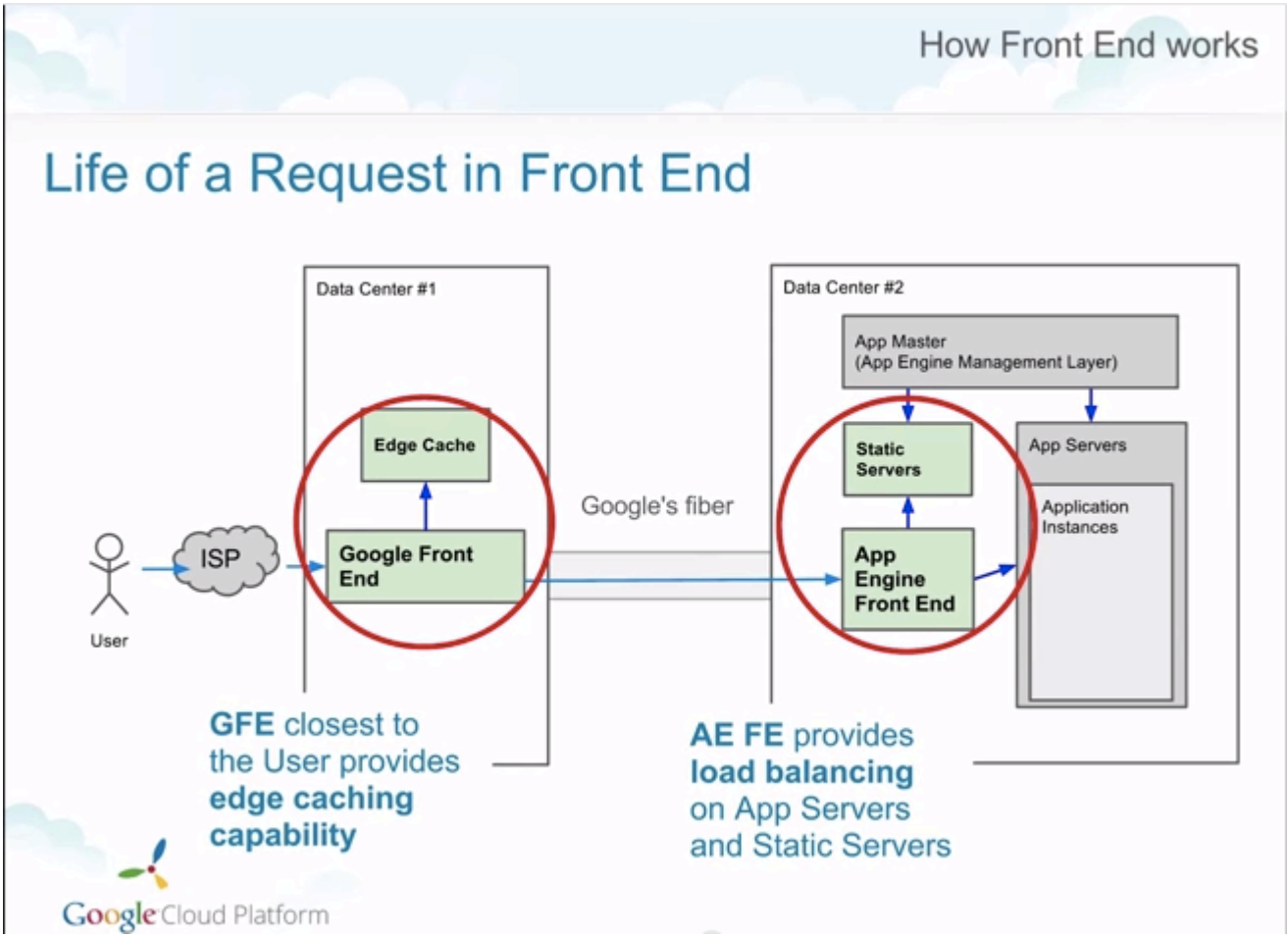
Détails

- Lorsque le cache Edge est activé et fonctionne, App Engine envoie un en-tête `age` dont la valeur est la durée (en secondes) depuis laquelle la réponse a été mise en cache. Par exemple, si la réponse a été mise en cache pendant deux minutes à ce jour, la réponse inclura un en-tête d' `age: 120` . Si aucun en-tête d' `age` n'est envoyé, c'est un bon signe que vous n'avez pas encore utilisé le cache Edge. De plus, lorsque la réponse n'inclut pas d'en-tête d' `age` , cela signifie que le cache Edge de la requête a été ignoré.
- Edge Cache ne fonctionne pas dans l'environnement de développement local.
- Il existe différents caches pour différentes régions. Il existe des caches séparés pour les différentes régions du monde. Par exemple, une demande provenant d'Europe peut être envoyée dans le cache, tandis qu'une demande à la même heure en provenance d'Australie peut ne pas atteindre le cache Edge.
- Il existe plusieurs caches même dans la même région. Par exemple, deux requêtes séquentielles provenant du même client peuvent recevoir des réponses avec des valeurs différentes pour l'en-tête `age` .
- Google expulse le contenu du cache Edge avant que l'âge maximal ne soit atteint. Cela est particulièrement vrai si la ressource n'a pas été demandée depuis plus de 5 minutes.
- Google envoie automatiquement un en-tête `Vary` avec la valeur `Accept-Encoding` avec certaines réponses. Voir cette documentation .
- App Engine ne vous permettra pas de définir l'en `Cache-Control` tête `Cache-Control` sur public si vous avez également un en `Set-Cookie` tête `Set-Cookie` présent. App Engine va même changer la valeur de l'en `Cache-Control` tête `Cache-Control` de `public` à `private` s'il existe un en `Set-Cookie` tête `Set-Cookie` . Voir ici

Comment fonctionne EdgeCache

EdgeCache est un cache de proxy inverse qui stocke les données pendant un certain temps et les renvoie rapidement dès qu'elles voient la même demande tant que le cache est toujours valide.

Voici un diagramme de cette vidéo de l'équipe App Engine sur le fonctionnement de EdgeCache:



Les centres de données "Google Front End" sont situés dans le monde entier et peuvent stocker des données mises en cache pour être renvoyées rapidement sur demande sans avoir à exécuter aucun de vos codes App Engine (qui s'exécutent sur le "Front App Engine").

Voir cette réponse StackOverflow pour plus d'informations sur les proxys inversés en général.

Situation actuelle

Malheureusement, l'état actuel des connaissances sur EdgeCache de GAE est plutôt mauvais. L'étendue de la documentation sur cette fonctionnalité secrète peut être trouvée dans ce post du forum (lisez-le maintenant!) À partir de 2011 et les 24 secondes entre 11h11 et 11h35 dans cette vidéo de Google.

Davantage de ressources

Vous trouverez ci-dessous la liste des ressources supplémentaires que nous avons trouvées concernant la fonctionnalité EdgeCache d'App Engine:

- Vidéo sur l'architecture et les services App Engine
- Message de forum fantastique de Brandon Wirtz
- Google App Engine Issue # 2258 (ouverte depuis 2009)
- Question StackOverflow: Détails sur le proxy de mise en cache de Google App Engine?
- Un mot sur la mise en cache de moteur d'application
- Configuration de EdgeCache

Examples

Activation de EdgeCache

- Pour activer EdgeCache, définissez les en-têtes de réponse HTTP suivants (*ne déviez pas de ce format exact*):
  - `Cache-Control` tête `Cache-Control` à `public, max-age=X` où:
    - `X = the number of seconds that you want the response to be cached for`
    - `X > 60 seconds`
    - `X < 365*24*60*60`
  - Définissez l'en-tête `Pragma` sur `Public` .

Lire EdgeCache en ligne: https://riptutorial.com/fr/google-app-engine/topic/1827/edgecache

Chapitre 5: Exemples d'exécution Python pour Google Appengine

Examples

NDB avec Python sur AppEngine

NDB relie les modèles en tant qu'objets python, qui peuvent être stockés et accessibles dans le magasin de données Appengine NoSQL , disponible pour toutes les applications AppEngine.

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)
```

Sans les modèles définis, nous pouvons créer de nouveaux objets pour l'entrée dans le magasin de données:

```
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.
```

Ajouter un épisode et le relier à une série:

```
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
```

Plus tard, pour récupérer toutes les séries:

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

Des filtres pourraient être appliqués si tous les spectacles n'étaient pas souhaités.

Plus de lecture:

- Vie d'un magasin de données

Lire Exemples d'exécution Python pour Google Appengine en ligne: https://riptutorial.com/fr/google-app-engine/topic/5902/exemples-d-execution-python-pour-google-appengine

Chapitre 6: Test unitaire avec datastore

Examples

Créez un contexte avec un magasin de données fortement cohérent.

Lorsque vous testez avec la bibliothèque de tests de Google App Engine, les défis de la cohérence éventuelle sont présents de la même manière qu'ils seront en production. Par conséquent, pour écrire quelque chose dans le magasin de données afin de tester la récupération, vous devez créer un contexte qui soit fortement cohérent.

```
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.")
        }
    }
}
```

Lire Test unitaire avec datastore en ligne: https://riptutorial.com/fr/google-app-

engine/topic/6587/test-unitaire-avec-datastore

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec google-app-engine	4444, Ari Molzer, Chandana, Community, Dan O'Boyle, Edward Fung, Ezequiel Jadib, Harshal Patil, Paritosh Walvekar
2	Démarrage rapide avec les utilisateurs API Python, authentification App Engine	Nicolas Bortolotti
3	Démarrage rapide de Google App Engine pour Java	Chandana
4	EdgeCache	Ani, michaelrbock
5	Exemples d'exécution Python pour Google Appengine	Dan O'Boyle
6	Test unitaire avec datastore	MrWizard54