EBook Gratis

APRENDIZAJE

google-app-engine

Free unaffiliated eBook created from
Stack Overflow contributors.

#google-app-engine

Tabla de contenido

Acerca de

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

Capítulo 1: Empezando con google-app-engine

Observaciones

Hay dos formas de obtener el SDK de GAE (el SDK de GAE independiente frente a `gcloud` Google Cloud SDK). Hay pequeñas diferencias cuando se implementa la aplicación utilizando `gcloud` . Si está utilizando `gcloud` , puede usar la `gcloud app deploy ~/my_app/app.yaml` . El comportamiento es diferente de usar el antiguo `appcfg.py` . Si prefiere usar `appcfg.py` , encontrará que no está disponible. Es porque, por alguna razón, Google decidió ocultarlo del desarrollador. Sin embargo, `appcfg.py` está, de hecho, instalado en el directorio `google-cloud-sdk/platform/google_appengine/` . También hay otros scripts útiles, como `bulkloader.py` .

Versiones

Versión	Fecha de lanzamiento
1.9.40	2016-07-15

Examples

Preparar

Google AppEngine (GAE) es una plataforma como servicio (PaaS) que brinda la capacidad de implementar aplicaciones en "Google Scale". Es uno de los muchos servicios en Google Cloud Platform (GCP). Los desarrolladores pueden integrar otros servicios como Google Cloud Storage (GCS) y Google Cloud SQL en GCP fácilmente. Los desarrolladores pueden escribir un conjunto de código que se ejecuta localmente y se puede implementar fácilmente en Google Cloud Platform .

Una vista de pájaro de comenzar con AppEngine incluye lo siguiente:

- Instale el SDK para su idioma preferido (Go, Python, Java, PHP, Node.js (en Beta))
- Usa el SDK para crear una aplicación y desarrollarla localmente
- Implemente el mismo código que se ejecuta localmente, en un entorno de tiempo de ejecución escalable

El SDK de AppEngine también se puede instalar utilizando el SDK de Google Cloud:

- Instale el SDK de Google Cloud
- Inicializa el Google Cloud SDK
- Autoriza el Google Cloud SDK
- Instale los componentes GAE. Para el usuario de Python,

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

y para el usuario go,

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

Para otros idiomas, use la `gcloud components list` para obtener la lista de componentes instalados y disponibles.

- Después de un tiempo, se instalará el SDK de GAE.

Otros enlaces útiles:

- Documentación formal de Google

Lea Empezando con google-app-engine en línea: https://riptutorial.com/es/google-app-engine/topic/971/empezando-con-google-app-engine

Capítulo 2: EdgeCache

Observaciones

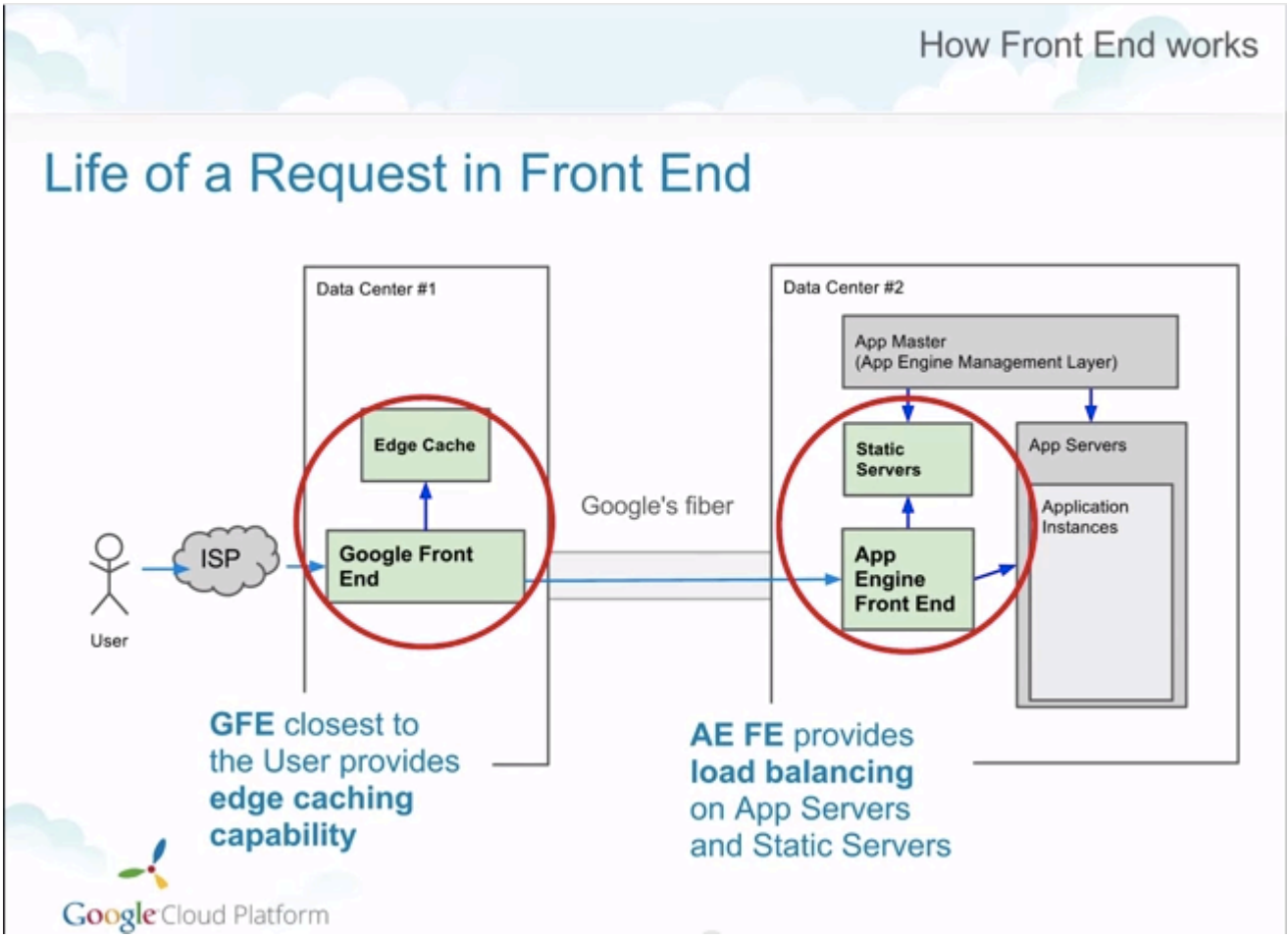
Detalles

- Cuando Edge Cache está habilitado y funcionando, App Engine envía un encabezado de `age` cuyo valor es el tiempo (en segundos) desde que la respuesta se almacenó en caché. Por ejemplo, si la respuesta se ha almacenado en caché durante dos minutos hasta ahora, la respuesta incluirá un encabezado de `age: 120` . Si no se envía un encabezado de `age` , es una buena señal de que aún no ha conseguido que Edge Cache funcione. Además, cuando la respuesta no incluye un encabezado de `age` , significa que la solicitud perdió la memoria caché perimetral.
- Edge Cache no funciona en el entorno de desarrollo local.
- Hay diferentes cachés para diferentes regiones. Hay cachés separados para regiones separadas del mundo. Por ejemplo, una solicitud de Europa puede llegar a la memoria caché, mientras que una solicitud en el mismo momento de Australia puede no llegar a Edge Cache.
- Hay múltiples cachés incluso en la misma región. Por ejemplo, dos solicitudes secuenciales del mismo cliente pueden recibir respuestas con valores diferentes para el encabezado de `age` .
- Google desaloja el contenido de Edge Cache antes de que se alcance la edad máxima. Esto es especialmente cierto si el recurso no se ha solicitado durante más de 5 minutos.
- Google envía automáticamente un encabezado `Vary` con el valor `Accept-Encoding` con algunas respuestas. Consulte esta documentación .
- App Engine no te permitirá configurar el encabezado de `Cache-Control` en público si también tienes un encabezado `Set-Cookie` presente. App Engine incluso cambiará el valor del encabezado `Cache-Control` de `public` a `private` si hay un encabezado `Set-Cookie` . Ver aqui

Cómo funciona EdgeCache

EdgeCache es un caché de proxy inverso que almacena datos durante un cierto período de tiempo y los devuelve rápidamente al ver la misma solicitud, siempre y cuando el caché aún sea válido.

Aquí hay un diagrama de este video del equipo de App Engine sobre cómo funciona EdgeCache:



Los centros de datos de "Google Front End" están ubicados en todo el mundo y pueden almacenar datos almacenados en caché para que se devuelvan rápidamente a pedido sin tener que ejecutar ningún código de App Engine (que se ejecuta en el "Front End de App Engine").

Consulte esta respuesta de StackOverflow para obtener más información acerca de los proxies inversos en general.

Situación actual

Desafortunadamente, el estado actual del conocimiento sobre EdgeCache de GAE es bastante malo. La extensión de la documentación sobre esta característica secreta se puede encontrar en esta publicación del foro (¡léala ahora!) De 2011 y los 24 segundos entre las 11:11 y las 11:35 en este video de Google.

Más recursos

A continuación se muestra la lista de más recursos que hemos encontrado relacionados con la función EdgeCache de App Engine:

- Video de App Engine Architecture and Services
- Fantástico post del foro de Brandon Wirtz
- Número de Google App Engine # 2258 (que ha estado abierto desde 2009)
- Pregunta de StackOverflow: ¿ Detalles sobre el proxy de almacenamiento en caché de Google App Engine?
- Una palabra en App Engine Caching

- Configurando EdgeCache

Examples

Habilitando EdgeCache

- Para habilitar EdgeCache, configure los siguientes encabezados de respuesta HTTP (*No se desvíe de este formato exacto*):
    - `Cache-Control` a `public, max-age=X` donde:
        - `X = the number of seconds that you want the response to be cached for`
        - `X > 60 seconds`
        - `X < 365*24*60*60`
    - Establecer el encabezado `Pragma` en `Public` .

Lea EdgeCache en línea: https://riptutorial.com/es/google-app-engine/topic/1827/edgecache

Capítulo 3: Ejemplos de Python Runtime para Google Appengine

Examples

NDB con Python en AppEngine

NDB relaciona los modelos como objetos de Python, que se pueden almacenar y acceder en el almacén de datos de Appengine NoSQL , disponible para todas las aplicaciones de AppEngine.

modelos.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)
```

Sin nuestros modelos definidos, podemos crear nuevos objetos para ingresar al almacén de datos:

```
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.
```

Añadiendo un episodio y relacionándolo con una serie:

```
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
```

Más tarde, para recuperar todas las series:

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

Se podrían aplicar filtros si no se desearan todos los shows.

Más lectura:

- La vida de un almacén de datos

Lea Ejemplos de Python Runtime para Google Appengine en línea: https://riptutorial.com/es/google-app-engine/topic/5902/ejemplos-de-python-runtime-para-google-appengine

Capítulo 4: Google App Engine Quickstart para Java

Examples

Antes de que empieces

Antes de ejecutar este ejemplo, debe:

Descargue e instale el Kit de desarrollo de Java SE (JDK):

Descargar JDK

Descargue la versión 3.3.9 o superior de Apache Maven :

Instale y configure Maven para su entorno de desarrollo local.

Descarga la aplicación Hello World

Hemos creado una sencilla aplicación Hello World para Java para que pueda familiarizarse rápidamente con la implementación de una aplicación en Google Cloud Platform. Siga estos pasos para descargar Hello World en su máquina local.

Clone el repositorio de la aplicación de ejemplo Hello World en su máquina local:

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

Vaya al directorio que contiene el código de muestra:

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

En los archivos `helloworld` resultantes encontrará el directorio `src` para un paquete llamado `com.example.appengine.helloworld` que implementa un simple `HTTPServlet` .

Alternativamente, puede descargar la muestra como un archivo zip y extraerlo.

Probar la aplicación

Pruebe la aplicación utilizando el servidor web de desarrollo, que se incluye con el SDK de App Engine.

1. Ejecute el siguiente comando de Maven desde su directorio `helloworld` para compilar su aplicación e iniciar el servidor web de desarrollo:

   mvn appengine: devserver

El servidor web ahora está escuchando las solicitudes en el puerto 8080.

2. Visite http: // localhost: 8080 / en su navegador web para ver la aplicación en acción.

Para obtener más información sobre cómo ejecutar el servidor web de desarrollo, consulte la referencia del servidor de desarrollo de Java .

Hacer un cambio

Puede dejar el servidor web en ejecución mientras desarrolla su aplicación. Cuando realice un cambio, use el `mvn clean package` para compilar y actualizar su aplicación.

1. Pruébelo ahora: deje el servidor web en ejecución, luego edite `HelloServlet.java` para cambiar `Hello, world` a algo más.
2. Ejecute `mvn clean package` , luego vuelva a cargar http: // localhost: 8080 / para ver los resultados.

Implementa tu aplicación

Para implementar su aplicación en App Engine, deberá registrar un proyecto para crear su ID de proyecto, que determinará la URL de la aplicación.

1. En la consola de la plataforma en la nube, vaya a la página Proyectos y seleccione o cree un nuevo proyecto.

2. Anote el ID del proyecto que creó anteriormente e ingréselo en src / main / webapp / WEB-INF / appengine-web.xml. También puede establecer la versión de la aplicación en este archivo.

3. Cargue su aplicación a Google App Engine invocando el siguiente comando.

   mvn appengine: actualización

4. Su aplicación ahora está implementada y lista para servir tráfico en http: // <YOUR_PROJECT_ID> .appspot.com /.

Lea Google App Engine Quickstart para Java en línea: https://riptutorial.com/es/google-app-engine/topic/6831/google-app-engine-quickstart-para-java

Capítulo 5: Inicio rápido con la API de Python de los usuarios, autenticación de App Engine

Introducción

El uso de la API de usuarios es una forma muy simple y flexible de trabajar con la autenticación en App Engine, pero asegúrese de que los casos de su aplicación no requieran más elementos para el entorno de autenticación.

Nota: Si necesita más información sobre la estructura tradicional de una aplicación de App Engine, revise esta información .

Observaciones

La API de Usuarios permite:

- Detectar si el usuario actual ha iniciado sesión.
- Redirige al usuario a la página de inicio de sesión correspondiente para iniciar sesión.
- Solicite al usuario de la aplicación que cree una nueva cuenta de Google si aún no la tiene.

Referencia y más detalles.

Elementos importantes en la vista:

Importar:

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

Usuario-objeto y métodos:

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

Nota: la implementación de jinja2 es opcional, pero en el artículo se utiliza para explicar el flujo de trabajo completo.

Examples

Controlador de página principal [views.py]

Importaciones generales, utilizando jinja2 para rellenar plantillas en htmls.

```
import jinja2
import webapp2
```

Importación importante para usar la API de usuarios:

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

Configuración del entorno Jinja: [en el ejemplo, la tecnología seleccionada para rellenar la información en el frontend]

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

Manipulador de hormigón:

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

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

Puede incluir más lógica aquí para los *usuarios*

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

Plantillas para pasar información utilizando jinja2. Para este ejemplo, el objeto de usuario y la cadena url.

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

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

Usando el ejemplo de index.html. [página html tradicional]

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

Enrutamiento de aplicaciones [urls.py]

Utilicé para este ejemplo webapp2 para cubrir el enrutamiento.

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

Importar desde vistas:

```
from views import MainPage
```

MainPage es el controlador establecido en la raíz "/":

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

Html, ejemplo de frontend de cómo usar la API de usuarios [index.html]

Extracto simple de index.html:

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

        {% if user %}
```

[Al pasar la url tenemos la oportunidad de cerrar la sesión del usuario]

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

[Puede incluir aquí las operaciones para el usuario autenticado]

```
        {% else %}
```

[Al pasar la url tenemos la oportunidad de iniciar sesión en el usuario]

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

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

Este es un ejemplo simple de cómo se utiliza la operación en la página index.html.

Lea Inicio rápido con la API de Python de los usuarios, autenticación de App Engine en línea: https://riptutorial.com/es/google-app-engine/topic/10002/inicio-rapido-con-la-api-de-python-de-los-usuarios--autenticacion-de-app-engine

Capítulo 6: Pruebas unitarias con almacén de datos

Examples

Crear un contexto con un almacén de datos muy consistente.

Al realizar pruebas con la biblioteca de pruebas de Google App Engine, los desafíos de la consistencia eventual están presentes de la misma manera que estarán en producción. Por lo tanto, para escribir algo en el almacén de datos para probar la recuperación, debe crear un contexto que sea muy consistente.

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

Lea Pruebas unitarias con almacén de datos en línea: https://riptutorial.com/es/google-app-engine/topic/6587/pruebas-unitarias-con-almacen-de-datos

Creditos

S. No	Capítulos	Contributors
1	Empezando con google-app-engine	4444, Ari Molzer, Chandana, Community, Dan O'Boyle, Edward Fung, Ezequiel Jadib, Harshal Patil, Paritosh Walvekar
2	EdgeCache	Ani, michaelrbock
3	Ejemplos de Python Runtime para Google Appengine	Dan O'Boyle
4	Google App Engine Quickstart para Java	Chandana
5	Inicio rápido con la API de Python de los usuarios, autenticación de App Engine	Nicolas Bortolotti
6	Pruebas unitarias con almacén de datos	MrWizard54