FREE eBook

LEARNING google-app-engine

Free unaffiliated eBook created from **Stack Overflow contributors.**

#google-app-engine

Table of Contents

About

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

Chapter 1: Getting started with google-app-engine

Remarks

There are two ways to get the GAE SDK (the standalone GAE SDK vs Google Cloud SDK `gcloud`). There are slight differences when the deploying the app using `gcloud`. If you are using `gcloud`, you can use `gcloud app deploy ~/my_app/app.yaml`. The behaviour is different from using the old `appcfg.py`. If you prefer using `appcfg.py`, you will find that it is not available. It is because for some reasons, Google decided to hide it from the developer. However, the `appcfg.py` is, in fact, installed in the directory `google-cloud-sdk/platform/google_appengine/`. Other useful scripts such as `bulkloader.py` are there as well.

Versions

Version	Released Date
1.9.40	2016-07-15

Examples

Setup

Google AppEngine (GAE) is a Platform as a Service (PaaS) that provides the ability to deploy applications at "Google Scale". It is one of the many services on Google Cloud Platform (GCP). Developers can integrate other services such as Google Cloud Storage (GCS) and Google Cloud SQL on GCP easily. Developers can write a set of code that runs locally and can easily be deployed on Google Cloud Platform.

A birds eye view of getting started with AppEngine includes the following:

- Install the SDK for your preferred language (Go, Python, Java, PHP, Node.js(in Beta))
- Use the SDK to scaffold an application & develop locally
- Deploy the same code that runs locally, to a scalable runtime environment

The AppEngine SDK can also be installed using the Google Cloud SDK:

- Install the Google Cloud SDK
- Initialize the Google Cloud SDK
- Authorize the Google Cloud SDK
- Install the GAE components. For python user,

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

and for go user,

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

For other languages, use `gcloud components list` to get the list of installed and available components.

- After awhile, the GAE SDK will be installed.

Other useful links:

- Formal Google Documentation

Read Getting started with google-app-engine online: https://riptutorial.com/google-app-engine/topic/971/getting-started-with-google-app-engine

Chapter 2: EdgeCache

Remarks

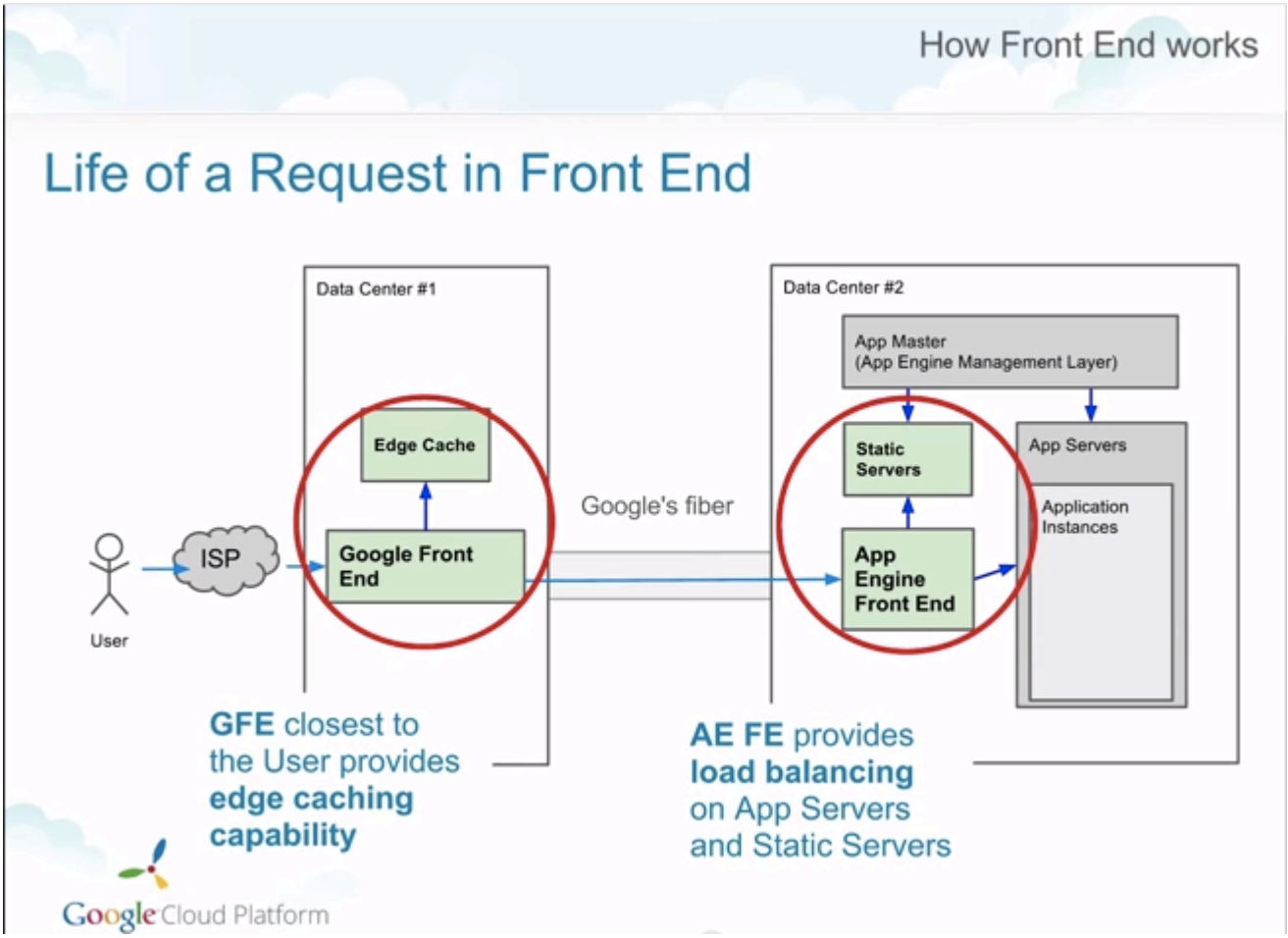
Details

- When Edge Cache is enabled and working, App Engine sends an `age` header whose value is the time (in seconds) since the response has been cached. For example, if the response has been cached for two minutes thus far, the response will include a header of `age: 120`. If no `age` header is being sent, it's a good sign that you haven't gotten Edge Cache to work yet. Additionally, when the response doesn't include an `age` header, it means the request missed Edge Cache.
- Edge Cache doesn't work in the local development environment.
- There are different caches for different regions. There are separate caches for separate regions of the world. For example, a request from Europe might be hit the cache, while a request at the exact same time from Australia may not hit Edge Cache.
- There are multiple caches even in the same region. For example, two sequential requests from the same client may receive responses with different values for the `age` header.
- Google evicts content from Edge Cache before the max-age has been reached. This is especially true if the resource has not been requested for more than 5 minutes.
- Google automatically sends a `Vary` header with the value `Accept-Encoding` with some responses. See this documentation.
- App Engine will not allow you to set the `Cache-Control` header to public if you also have a `Set-Cookie` header present. App Engine will even change the value of the `Cache-Control` header from `public` to `private` if there is a `Set-Cookie` header. See here.

How EdgeCache works

EdgeCache is a reverse proxy cache that stores data for a certain period of time, and returns it quickly upon seeing the same request as long as the cache is still valid.

Here is a diagram from this video from the App Engine team about how EdgeCache works:



The "Google Front End" data centers are located around the world and can store cached data to be returned quickly upon request without ever having to run any of your App Engine code (which runs on the "App Engine Front End").

See this StackOverflow answer for more information about reverse proxies in general.

Current Situation

Unfortunately, the current state of knowledge about GAE's EdgeCache is pretty bad. The extent of the documentation on this secretive feature can be found in this forum post (read it now!) from 2011 and the 24 seconds between 11:11 and 11:35 in this video from Google.

More Resources

Below is the list of more resources we've found pertaining to App Engine's EdgeCache feature:

- App Engine Architecture and Services video
- Fantastic forum post from Brandon Wirtz
- Google App Engine Issue #2258 (which has been open since 2009)
- StackOverflow question: Details on Google App Engine's caching proxy?
- A Word on App Engine Caching
- Configuring EdgeCache

Examples

Enabling EdgeCache

- In order to enable EdgeCache, set the following HTTP response headers (*Do not deviate from this exact format*):
  - `Cache-Control` header to `public, max-age=X` where:
    - `X = the number of seconds that you want the response to be cached for`
    - `X > 60 seconds`
    - `X < 365*24*60*60`
  - Set the `Pragma` header to `Public`.

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

Chapter 3: Google App Engine Quickstart for Java

Examples

Before you begin

Before running this sample, you must:

Download and install the Java SE Development Kit (JDK):

Download JDK

Download Apache Maven version 3.3.9 or greater:

Install and configure Maven for your local development environment.

Download the Hello World app

We've created a simple Hello World app for Java so you can quickly get a feel for deploying an app to Google Cloud Platform. Follow these steps to download Hello World to your local machine.

Clone the Hello World sample app repository to your local machine:

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

Go to the directory that contains the sample code:

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

In the resulting `helloworld` files you'll find the `src` directory for a package called `com.example.appengine.helloworld` that implements a simple `HTTPServlet`.

Alternatively, you can download the sample as a zip file and extract it.

Test the application

Test the application using the development web server, which is included with the App Engine SDK.

1. Run the following Maven command from within your `helloworld` directory, to compile your app and start the development web server:

   mvn appengine:devserver

The web server is now listening for requests on port 8080.

2. Visit http://localhost:8080/ in your web browser to see the app in action.

For more information about running the development web server, see the Java Development Server reference.

Make a change

You can leave the web server running while you develop your application. When you make a change, use the `mvn clean package` command to build and update your app.

1. Try it now: Leave the web server running, then edit `HelloServlet.java` to change `Hello, world` to something else.
2. Run `mvn clean package`, then reload http://localhost:8080/ to see the results.

Deploy your app

To deploy your app to App Engine, you will need to register a project to create your project ID, which will determine the URL for the app.

1. In the Cloud Platform Console, go to the Projects page and select or create a new project.

2. Note the project ID you created above, and enter it in src/main/webapp/WEB-INF/appengine-web.xml. You can also set the app version in this file.

3. Upload your application to Google App Engine by invoking the following command.

   mvn appengine:update

4. Your app is now deployed and ready to serve traffic at http://<YOUR_PROJECT_ID>.appspot.com/.

Read Google App Engine Quickstart for Java online: https://riptutorial.com/google-app-engine/topic/6831/google-app-engine-quickstart-for-java

Chapter 4: Python Runtime Examples for Google Appengine

Examples

NDB with Python on AppEngine

NDB relates models as python objects, which can be stored and accessed in the Appengine NoSQL datastore, available to all AppEngine applications.

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)
```

With out models defined we can create new objects for entry to the datastore:

```
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.
```

Adding an episode and relating it to a Series:

```
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
```

Later, to retrieve all Series:

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

Filters could be applied if all shows were not desired.

More reading:

- Life of a Datastore Write

Read Python Runtime Examples for Google Appengine online: https://riptutorial.com/google-app-engine/topic/5902/python-runtime-examples-for-google-appengine

Chapter 5: Quick start with Users Python API, App Engine Authentication

Introduction

Using the Users API is a very simple and flexible way to work the authentication in App Engine, but please make sure that your application cases don't require more elements for the authentication environment.

Note: If you need more information about the traditional structure of an App Engine app, please review this info.

Remarks

The Users API allows:

- Detect whether the current user has signed in.
- Redirect the user to the appropriate sign-in page to sign in.
- Request that your application user create a new Google account if they don't have one already.

Reference and more details

Important elements into the view:

Import:

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

User-object and methods:

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

Note: the implementation of jinja2 is optional, but into the article is used to explaining the completely workflow.

Examples

MainPage Handler [views.py]

General Imports, using jinja2 to populate templates into htmls.

```
import jinja2
import webapp2
```

Important import to use Users API:

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

Setting of Jinja environment: [into the example the tehcnology selected to populate the information into the frontend]

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

Concrete Handler:

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

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

You can include more logic here for *users*

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

Templates to pass information using jinja2. For this example, the user object and the url string.

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

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

Using index.html example. [traditional html page]

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

App Routing [urls.py]

I used for this example webapp2 to cover the routing.

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

Import from views:

```
from views import MainPage
```

MainPage is the handler set into root "/":

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

Html, frontend example of how use Users API [index.html]

Simple extract of index.html:

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

        {% if user %}
```

[Passing the url we have the opportunity to logout the user]

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

[You can include here operations for user authenticated]

```
        {% else %}
```

[Passing the url we have the opportunity to login the user]

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

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

This is a simple example of how the operation is used on the index.html page.

Read Quick start with Users Python API, App Engine Authentication online: https://riptutorial.com/google-app-engine/topic/10002/quick-start-with-users-python-api--app-engine-authentication

Chapter 6: Unit testing with datastore

Examples

Create a context with a strongly consistent data store.

When testing with Google App Engine's testing library the challenges of eventual consistency are present in the same manner they will be in production. Therefore in order to write something into the datastore to test retrieval you have to create a context which is strongly consistent.

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

Read Unit testing with datastore online: https://riptutorial.com/google-app-engine/topic/6587/unit-testing-with-datastore

Credits

S. No	Chapters	Contributors
1	Getting started with 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 Quickstart for Java	Chandana
4	Python Runtime Examples for Google Appengine	Dan O'Boyle
5	Quick start with Users Python API, App Engine Authentication	Nicolas Bortolotti
6	Unit testing with datastore	MrWizard54