



Building apps with GAE

Deploying an app to Google App Engine, using python and the webapp2 framework



Prerequisites

Download and configure the Google App Engine Python SDK from:

<https://cloud.google.com/appengine/downloads>

You can also download all the code and slides for this workshop at:

https://github.com/mjmccconnell/gae_tutorial



Step one

The app.yaml

app.yaml

This is your main configuration file for the app

This file specifies how URL paths correspond to request handlers and static files. It also contain information about the application code, such as application ID, runtime (in this case Python) and version

This and a static file is all you need to have a live website

The application ID, is the ID you get when creating your app at <https://console.developers.google.com>

You can then deploy your app with `appcfg.py update` .



Step two

Here comes the Python

main.py

Instead of using static files to render content to our web app, we can use python to write to the response.

This means that we can use dynamic content, such as Python's "date" object.

We then serve these new python methods to write the response to a given url.

No need for any HTML, CSS or JS



Mixing Python with HTML

Jinja

GAE comes with a host of third-party libraries ready for you to use. One of the most commonly used (in any python web app), is Jinja.

Jinja is a templating language that allows easy python integration inside static files, such as HTML.

Its syntax is fairly similar to that of Python, and can be identified by looking for curly braces { }.



Passing data around your app

GET vs POST

There are two methods of passing data around any web app, in any language, the GET and POST method.

Both methods can do the same job as each other, but shouldn't.

The GET method should be used to retrieve data.

Example: www.yoursite.com?page=1 which would in theory return data for page 1

The POST method should be used to submit data

Example: A guest book form, where a user “submits” data to be saved somewhere



Persisting data

The NDB Datastore

Google App Engine comes with its own “database”, which can be used to store data persistently.

At its most basic, write and reading from the datastore is very simple, as it requires no setup, you can just start writing to it.



Cleaning up

Inheritance and Importing

Currently all of our python code is in one file (the main.py), but as we expand out the functionality of our site, this will start to bloat out fast, and is not really scalable.

So we can use “imports” to refactor out parts of our main.py into seperate files and folders.

We can also chop of parts of the code we use more than once, and share a single “method” between “classes”