

Apr 24, 18 23:07

taxi-handler.yaml

Page 1/1

```

1 # -----
2
3 service: taxi-handler
4 runtime: python27
5 api_version: 1
6 threadsafe: true
7
8 handlers:
9
10 - url: /registerTaxi
11   script: registerTaxi.app
12
13 - url: /startTrip
14   script: startTrip.app
15
16 - url: /endTrip
17   script: endTrip.app
18
19 - url: /checkFacturacion
20   script: checkFacturacion.app
21
22 libraries:
23
24 - name: webapp2
25   version: latest
26
27 - name: jinja2
28   version: latest

```

Apr 24, 18 17:43

startTrip.py

Page 1/1

```

1
2 # [START imports]
3 import os
4 import urllib
5 from urlparse import *
6 from datetime import datetime
7
8 from google.appengine.ext import ndb
9 from Models import *
10
11 import webapp2
12 from webapp2_extras import json
13
14 # [START startTrip]
15 class StartTrip(webapp2.RequestHandler):
16
17     def get(self):
18
19         driver_id = self.request.get("driverId")
20         driver_key = ndb.Key(urlsafe=driver_id)
21         new_id = ndb.Model.allocate_ids(size=1, parent=driver_key)[0]
22         trip_key = ndb.Key('Trip', new_id, parent=driver_key)
23
24         pickup_datetime = self.request.get("datetime")
25         pickup_latitude = self.request.get("latitude")
26         pickup_longitude = self.request.get("longitude")
27
28         trip = Trip(key=trip_key,
29                     pickup_datetime=datetime.strptime(pickup_datetime, "%Y-%m-%d %H:%M:%S"),
30                     pickup_location=ndb.GeoPt(pickup_latitude, pickup_longitude))
31
32         trip.put();
33         self.response.content_type = 'application/json'
34         response_json = {
35             'success': 'OK',
36             'tripKey': trip_key.urlsafe()
37         }
38         self.response.write(json.encode(response_json))
39
40 # [START app]
41 app = webapp2.WSGIApplication([
42     ('/startTrip', StartTrip)
43 ], debug=True)
44 # [END app]

```

Apr 24, 18 17:43

registerTaxi.py

Page 1/1

```

1
2 # [START imports]
3 import os
4 import urllib
5 from urlparse import *
6 from datetime import datetime
7
8 from google.appengine.ext import ndb
9 from Models import *
10
11 #import jinja2
12 import webapp2
13 from webapp2_extras import json
14
15 # [START registerTaxi]
16 class RegisterTaxi(webapp2.RequestHandler):
17
18     def get(self):
19
20         driver_name = self.request.get("driverName")
21
22         taxi = Taxi(driver_name=driver_name)
23
24         taxi_key = taxi.put()
25         taxi_key_urlsafe = taxi_key.urlsafe()
26
27         self.response.content_type = 'application/json'
28         response_json = {
29             'success': 'OK',
30             'taxiKey': taxi_key_urlsafe
31         }
32         self.response.write(json.encode(response_json))
33
34 # [START app]
35 app = webapp2.WSGIApplication([
36     ('/registerTaxi', RegisterTaxi)
37 ], debug=True)
38 # [END app]

```

Apr 23, 18 17:10

queue.yaml

Page 1/1

```

1 queue:
2 - name: trips-per-day
3   mode: pull

```

Apr 24, 18 22:10

Models.py

Page 1/1

```

1  from google.appengine.ext import ndb
2
3  # [START taxi]
4  class Taxi(ndb.Model):
5      """A main model for representing a taxi driver."""
6      driver_name = ndb.StringProperty(indexed=True);
7      # [END taxi]
8
9  # [START trip]
10 class Trip(ndb.Model):
11     """A main model for representing a taxi trip."""
12     pickup_datetime = ndb.DateTimeProperty(indexed=True)
13     pickup_location = ndb.GeoPtProperty(indexed=False)
14     dropoff_datetime = ndb.DateTimeProperty(indexed=True)
15     dropoff_location = ndb.GeoPtProperty(indexed=False)
16     distance = ndb.FloatProperty(indexed=False)
17     # [END trip]
18
19     # [START statistic]
20 class DailyStatistic(ndb.Model):
21     """A main model for representing a daily trips statistic."""
22     date = ndb.DateProperty(indexed=True)
23     trips = ndb.IntegerProperty(indexed=False)
24     # [END statistic]

```

Apr 23, 18 3:13

index.yaml

Page 1/1

```

1  indexes:
2
3  # AUTOGENERATED
4
5  # This index.yaml is automatically updated whenever the dev_appserver
6  # detects that a new type of query is run. If you want to manage the
7  # index.yaml file manually, remove the above marker line (the line
8  # saying "# AUTOGENERATED"). If you want to manage some indexes
9  # manually, move them above the marker line. The index.yaml file is
10 # automatically uploaded to the admin console when you next deploy
11 # your application using appcfg.py.
12
13 - kind: Trip
14   ancestor: yes
15   properties:
16     - name: pickup_datetime
17       direction: desc

```

Apr 26, 18 16:53

endTrip.py

Page 1/1

```

1
2 # [START imports]
3 import os
4 import urllib
5 from urlparse import *
6 from datetime import datetime
7 import logging
8
9 from google.appengine.ext import ndb
10 from google.appengine.api import memcache
11 from google.appengine.api import datastore
12 from google.appengine.api import taskqueue
13 from Models import *
14 from Configuration import *
15
16 import webapp2
17 from webapp2_extras import json
18
19 # [START endTrip]
20 class EndTrip(webapp2.RequestHandler):
21
22     def get(self):
23
24         trip_id = self.request.get("tripId")
25         trip_key = ndb.Key(urlsafe=trip_id)
26
27         trip = memcache.get(trip_key.urlsafe())
28         if trip is None:
29             trip = trip_key.get()
30             if trip is None:
31                 logging.error("Received endTrip request for invalid trip id: %s", trip_id)
32             return
33         memcache.add(trip_key.urlsafe(), trip)
34
35         dropoff_datetime = self.request.get("datetime")
36         dropoff_latitude = self.request.get("latitude")
37         dropoff_longitude = self.request.get("longitude")
38         distance = float(self.request.get("distance"))
39
40         trip.dropoff_datetime = datetime.strptime(dropoff_datetime, "%Y-%m-%d %H:%M:%S")
41         trip.dropoff_location = ndb.GeoPt(dropoff_latitude, dropoff_longitude)
42         trip.distance = distance
43
44         trip.put();
45
46         # pass to pull queue for dayli trips counter
47         tag = trip.pickup_datetime.strftime("%Y-%m-%d")
48         q = taskqueue.Queue(Configuration.DailyTripsTaskQueue)
49         tasks = []
50         payload_str = '1'
51         tasks.append(taskqueue.Task(payload=payload_str, method='PULL', tag=tag))
52         q.add(tasks)
53
54         self.response.content_type = 'application/json'
55         response_json = {
56             'success': 'OK'
57         }
58         self.response.write(json.encode(response_json))
59
60 # [START app]
61 app = webapp2.WSGIApplication([
62     ('/endTrip', EndTrip)
63 ], debug=True)
64 # [END app]

```

Apr 24, 18 23:05

dispatch.yaml

Page 1/1

```

1 dispatch:
2
3 - url: "*/startTrip"
4   service: taxis-handler
5
6 - url: "*/registerTaxi"
7   service: taxis-handler
8
9 - url: "*/endTrip"
10  service: taxis-handler
11
12 - url: "*/checkFacturacion"
13   service: taxis-handler
14
15 - url: "*/tasks/daily-trip-summary"
16   service: daily-trip-summary
17
18 - url: "*/adminQuery"
19   service: admin-query
20
21 - url: "*/dailyTripsStatistic"
22   service: daily-trips-statistic

```

Apr 24, 18 22:50

default.py

Page 1/1

```
1
2 # [START imports]
3 import os
4 import urllib
5 from urlparse import *
6
7 import jinja2
8 import webapp2
9 from webapp2_extras import json
10
11 JINJA_ENVIRONMENT = jinja2.Environment(
12     loader=jinja2.FileSystemLoader(os.path.dirname(__file__)),
13     extensions=['jinja2.ext.autoescape'],
14     autoescape=True)
15 # [END imports]
16
17 # [START main]
18 class MainPage(webapp2.RequestHandler):
19
20     def get(self):
21
22         template = JINJA_ENVIRONMENT.get_template('default-index.html')
23         self.response.write(template.render())
24
25 # [END main]
26
27 # [START app]
28 app = webapp2.WSGIApplication([
29     ('/', MainPage)
30 ], debug=True)
31 # [END app]
```

Whoops

You are not supposed to be here! {% endautoescape %}

Apr 23, 18 17:41

daily-trip-summary.yaml

Page 1/1

```

1 # -----
2
3 service: daily-trip-summary
4 runtime: python27
5 api_version: 1
6 threadsafe: true
7
8 handlers:
9
10 - url: /tasks/daily-trip-summary
11   script: dailyTripSummary.app
12
13 libraries:
14 - name: webapp2
15   version: latest

```

Apr 26, 18 16:30

dailyTripSummary.py

Page 1/1

```

1
2 # [START imports]
3 import os
4 import urllib
5 from urlparse import *
6 from datetime import datetime
7 from Configuration import *
8
9 from google.appengine.ext import ndb
10 from google.appengine.api import memcache
11 from google.appengine.api import datastore
12 from google.appengine.api import taskqueue
13 from Models import *
14
15 import webapp2
16 import logging
17
18 # [START endTrip]
19 class DailyTripSummary(webapp2.RequestHandler):
20
21     def get(self):
22
23         q = taskqueue.Queue(Configuration.DailyTripsTaskQueue)
24         tasks = q.lease_tasks_by_tag(Configuration.TasksLeaseDurationSeconds, Configuration.AmountOfTasksPerLease)
25         logging.info(tasks)
26
27         numTrips = len(tasks)
28         if numTrips <= 0:
29             return
30
31         tag = tasks[0].tag
32         statistic_key = ndb.Key('DailyStatistic', tag)
33         statistic = memcache.get(tag)
34         if statistic is None:
35             statistic = statistic_key.get()
36             if statistic is None:
37                 statistic = DailyStatistic(key=statistic_key, date=datetime.strptime(tag, "%Y-%m-%d"), trips=0)
38                 memcache.add(tag, statistic)
39                 statistic.trips += numTrips
40                 statistic.put()
41
42         q.delete_tasks(tasks)
43
44 # [START app]
45 app = webapp2.WSGIApplication([
46     ('/tasks/daily-trip-summary', DailyTripSummary)
47 ], debug=True)
48 # [END app]

```

Apr 24, 18 22:13

daily-trips-statistic.yaml

Page 1/1

```

1 # -----
2
3 service: daily-trips-statistic
4 runtime: python27
5 api_version: 1
6 threadsafe: true
7
8 handlers:
9
10 - url: /dailyTripsStatistic
11   script: dailyTripsStatistic.app
12
13 libraries:
14 - name: webapp2
15   version: latest
16
17 - name: jinja2
18   version: latest

```

Apr 24, 18 22:23

dailyTripsStatistic.py

Page 1/1

```

1
2 # [START imports]
3 import os
4 import urllib
5 from urlparse import *
6 from datetime import datetime
7 from datetime import timedelta
8
9 from google.appengine.ext import ndb
10 from google.appengine.api import memcache
11 from Models import *
12
13 import jinja2
14 import webapp2
15 from webapp2_extras import json
16
17 JINJA_ENVIRONMENT = jinja2.Environment(
18     loader=jinja2.FileSystemLoader(os.path.dirname(__file__)),
19     extensions=['jinja2.ext.autoescape'],
20     autoescape=True)
21 # [END imports]
22
23 # [START checkFacturacion]
24 class DailyTripsStatistic(webapp2.RequestHandler):
25
26     def get(self):
27
28         template_values = {
29             'date': datetime.today().strftime('%Y-%m-%d')
30         }
31
32         template = JINJA_ENVIRONMENT.get_template('daily-trips-statistic-index.html')
33         self.response.write(template.render(template_values))
34
35     def post(self):
36
37         date = self.request.get("date")
38         statistic_key = ndb.Key('DailyStatistic', date)
39
40         statistic = memcache.get(statistic_key.urlsafe())
41         if statistic is None:
42             statistic = statistic_key.get()
43             if statistic is None:
44                 num_trips = 0
45             else:
46                 memcache.add(statistic_key.urlsafe(), statistic)
47                 num_trips = statistic.trips
48         else:
49             num_trips = statistic.trips
50
51         showTrips = True
52         template_values = {
53             'date': date,
54             'number_of_trips': num_trips,
55             'showTrips': showTrips
56         }
57
58         template = JINJA_ENVIRONMENT.get_template('daily-trips-statistic-index.html')
59         self.response.write(template.render(template_values))
60
61 # [END checkFacturacion]
62
63 # [START app]
64 app = webapp2.WSGIApplication([
65     ('/dailyTripsStatistic', DailyTripsStatistic)
66 ], debug=True)
67 # [END app]

```


Viajes totales por dia

{% if showTrips %} Number of trips: {{ number_of_trips }} {% endif %} {% endautoescape %}

Apr 25, 18 18:01

cron.yaml

Page 1/1

```
1 cron:
2 - description: "daily trips summary"
3   url: /tasks/daily-trip-summary
4   target: daily-trip-summary
5   schedule: every 8 hours
```

Apr 26, 18 16:30

Configuration.py

Page 1/1

```
1
2 class Configuration():
3     AmountOfTasksPerLease = 100
4     TasksLeaseDurationSeconds = 300
5
6     AdminQueryDefaultTripsPerPage = 10
7
8     DailyTripsTaskQueue = "trips-per-day"
```

Apr 24, 18 18:03

checkFacturacion.py

Page 1/1

```

1
2 # [START imports]
3 import os
4 import urllib
5 from urlparse import *
6 from datetime import datetime
7 from datetime import timedelta
8
9 from google.appengine.ext import ndb
10 from Models import *
11
12 import jinja2
13 import webapp2
14 from webapp2_extras import json
15
16 JINJA_ENVIRONMENT = jinja2.Environment(
17     loader=jinja2.FileSystemLoader(os.path.dirname(__file__)),
18     extensions=['jinja2.ext.autoescape'],
19     autoescape=True)
20 # [END imports]
21
22 # [START checkFacturacion]
23 class CheckFacturacion(webapp2.RequestHandler):
24
25     def get(self):
26
27         template_values = {
28             'facturacion_results': {}
29         }
30
31         template = JINJA_ENVIRONMENT.get_template('check-facturacion-index.html')
32         self.response.write(template.render(template_values))
33
34     def post(self):
35
36         driver_id = self.request.get("driverId")
37         driver_key = ndb.Key(urlsafe=driver_id)
38
39         query_date = datetime.strptime(self.request.get("query_date"), "%Y-%m-%d")
40
41         facturacion_query = Trip.query(ndb.AND(Trip.pickup_datetime ≥ query_date, Trip.pickup_datetime < query_date + timedelta(days=1)), ancestor=driver_key).order(
42             (-Trip.pickup_datetime))
43         facturacion_results = facturacion_query.fetch()
44
45         template_values = {
46             'facturacion_results': facturacion_results
47         }
48
49         template = JINJA_ENVIRONMENT.get_template('check-facturacion-index.html')
50         self.response.write(template.render(template_values))
51 # [END checkFacturacion]
52
53 # [START app]
54 app = webapp2.WSGIApplication([
55     ('/checkFacturacion', CheckFacturacion)
56 ], debug=True)
57 # [END app]

```

Resumen de Facturaci3n

{% for trip in facturacion_results %}
Pickup time: {{trip.pickup_datetime}}, Pickup location: {{trip.pickup_location}}, Dropoff time:
{{trip.dropoff_datetime}}, Dropoff location: {{trip.dropoff_location}}, Distance: {{trip.distance}}
{% endfor %}
{% endautoescape %}

Apr 24, 18 22:49

app.yaml

Page 1/1

```
1 runtime: python27
2 api_version: 1
3 threadsafe: true
4
5 handlers:
6
7 - url: "/"
8   script: default.app
9
10 libraries:
11 - name: webapp2
12   version: latest
13
14 - name: jinja2
15   version: latest
```

Apr 24, 18 17:59

admin-query.yaml

Page 1/1

```
1 # -----
2
3 service: admin-query
4 runtime: python27
5 api_version: 1
6 threadsafe: true
7
8 handlers:
9
10 - url: /adminQuery
11   script: adminQuery.app
12
13 libraries:
14 - name: webapp2
15   version: latest
16
17 - name: jinja2
18   version: latest
```

Apr 26, 18 16:31

adminQuery.py

Page 1/2

```

1  # [START imports]
2  import os
3  import urllib
4  from urlparse import *
5  from datetime import datetime
6  from datetime import timedelta
7
8
9  from google.appengine.datastore.datastore_query import Cursor
10 from google.appengine.ext import ndb
11 from google.appengine.api import memcache
12 from Models import *
13 from Configuration import *
14
15 import jinja2
16 import webapp2
17 from webapp2_extras import json
18 import logging
19
20 JINJA_ENVIRONMENT = jinja2.Environment(
21     loader=jinja2.FileSystemLoader(os.path.dirname(__file__)),
22     extensions=['jinja2.ext.autoescape'],
23     autoescape=True)
24 # [END imports]
25
26 # [START adminQuery]
27 class AdminQuery(webapp2.RequestHandler):
28
29     def get(self):
30
31         template_values = {
32             'results': {},
33             'query_date_from': "2018-04-01",
34             'query_date_to': datetime.today().strftime('%Y-%m-%d'),
35             'driverId': "",
36             'prev_cursor': "",
37             'next_cursor': "",
38             'TRIPS_PER_PAGE': Configuration.AdminQueryDefaultTripsPerPage
39         }
40
41         template = JINJA_ENVIRONMENT.get_template('admin-query-index.html')
42         self.response.write(template.render(template_values))
43
44     def post(self):
45
46         TRIPS_PER_PAGE = int(self.request.get("TRIPS_PER_PAGE"))
47
48         # IF prev BUTTON WAS PRESSED
49         is_prev = self.request.get("prev", "") != ""
50
51         # DRIVER QUERY
52         driverId = self.request.get("driverId", "")
53
54         # DATE FROM QUERY
55         query_date_from = datetime.strptime(self.request.get("query_date_from"), "%Y-%m-%d")
56
57         # DATE TO QUERY
58         query_date_to = datetime.strptime(self.request.get("query_date_to"), "%Y-%m-%d")
59
60         # CURSOR
61         prev_cursor_before = self.request.get('prev_cursor', default_value="")
62         next_cursor_before = self.request.get('next_cursor', default_value="")
63
64         # if a driver_id was input, use it to filter
65         if driverId == "":
66             query = Trip.query(ndb.AND(Trip.pickup_datetime ≥ query_date_from, Trip.pickup_datetime < query_date_to + timedelta(days=1)))
67         else:
68             driver_key = ndb.Key(unsafe=driverId).get() # usar memcache
69             query = Trip.query(ndb.AND(Trip.pickup_datetime ≥ query_date_from, Trip.pickup_datetime < query_date_to + timedelta(days=1)), ancestor=driver_key)

```

Apr 26, 18 16:31

adminQuery.py

Page 2/2

```

70
71 # ----- PAGE MANAGEMENT
72 query_forward = query.order(Trip.pickup_datetime)
73 query_reverse = query.order(-Trip.pickup_datetime)
74
75 if is_prev:
76     qry = query_reverse
77     cursor = ndb.Cursor(unsafe=prev_cursor_before).reversed() if prev_cursor_before != "" else None
78 else:
79     qry = query_forward
80     cursor = ndb.Cursor(unsafe=next_cursor_before) if next_cursor_before != "" else None
81
82 # trips in page
83 query_results, cursor, more = qry.fetch_page(TRIPS_PER_PAGE, start_cursor=cursor)
84 # get driver info for each trip
85 results = []
86 drivers = {}
87 for result in query_results:
88     taxiKey = result.key.parent()
89     if taxiKey.unsafe() in drivers:
90         taxi = memcache.get(taxiKey.unsafe())
91     if taxi is None:
92         taxi = taxiKey.get()
93     if taxi is None:
94         logging.error("Received trip exists for taxiKey: %s, but taxi doesn't exist", trip_id)
95         taxi = { "driver_name": "MISSING" }
96     else:
97         memcache.add(taxiKey.unsafe(), taxi)
98         drivers[taxiKey.unsafe()] = taxi.driver_name
99     total_result = { "driver": drivers[taxiKey.unsafe()], "trip": result }
100    results.append(total_result)
101
102 if is_prev:
103     prev_cursor_url = cursor.reversed().unsafe() if more else ""
104     next_cursor_url = prev_cursor_before
105 else:
106     prev_cursor_url = next_cursor_before
107     next_cursor_url = cursor.unsafe() if more else ""
108 # -----
109
110 template_values = {
111     'results': results,
112     'query_date_from': query_date_from,
113     'query_date_to': query_date_to,
114     'driverId': driverId,
115     'prev_cursor': prev_cursor_url,
116     'next_cursor': next_cursor_url,
117     'TRIPS_PER_PAGE': TRIPS_PER_PAGE
118 }
119
120 template = JINJA_ENVIRONMENT.get_template('admin-query-index.html')
121 self.response.write(template.render(template_values))
122
123 # [END adminQuery]
124
125 # [START app]
126 app = webapp2.WSGIApplication([
127     ('/adminQuery', AdminQuery)
128 ], debug=True)
129 # [END app]

```

Acceso de administrador

trips per page:

```
{% for result in results %}
Driver name: {{result.driver}}, Pickup time: {{result.trip.pickup_datetime}}, Pickup location:
{{result.trip.pickup_location}}, Dropoff time: {{result.trip.dropoff_datetime}}, Dropoff location:
{{result.trip.dropoff_location}}, Distance: {{result.trip.distance}}
{% endfor %}

{% if prev_cursor != "" %} {% endif %} {% if next_cursor != "" %} {% endif %}

{% endautoescape %}
```

Apr 26, 18 18:46

Table of Content

Page 1/1

1	Table of Contents				
2	1	<i>taxis-handler.yaml</i> ..	sheets	1 to 1 (1)	pages 1- 1 29 lines
3	2	<i>startTrip.py</i>	sheets	1 to 1 (1)	pages 2- 2 45 lines
4	3	<i>registerTaxi.py</i>	sheets	2 to 2 (1)	pages 3- 3 39 lines
5	4	<i>queue.yaml</i>	sheets	2 to 2 (1)	pages 4- 4 4 lines
6	5	<i>Models.py</i>	sheets	3 to 3 (1)	pages 5- 5 25 lines
7	6	<i>index.yaml</i>	sheets	3 to 3 (1)	pages 6- 6 18 lines
8	7	<i>endTrip.py</i>	sheets	4 to 4 (1)	pages 7- 7 65 lines
9	8	<i>dispatch.yaml</i>	sheets	4 to 4 (1)	pages 8- 8 23 lines
10	9	<i>default.py</i>	sheets	5 to 5 (1)	pages 9- 9 32 lines
11	10	<i>default-index.html</i> ..	sheets	5 to 6 (2)	pages 9- 11 1 lines
12	11	<i>daily-trip-summary.yaml</i>	sheets	7 to 7 (1)	pages 12- 12 16 lines
13	12	<i>dailyTripSummary.py</i> .	sheets	7 to 7 (1)	pages 13- 13 49 lines
14	13	<i>daily-trips-statistic.yaml</i>	sheets	8 to 8 (1)	pages 14- 14 19 lines
15	14	<i>dailyTripsStatistic.py</i>	sheets	8 to 8 (1)	pages 15- 15 68 lines
16	15	<i>daily-trips-statistic-index.html</i>	sheets	8 to 9 (2)	pages 15- 17 1 lines
17	16	<i>cron.yaml</i>	sheets	10 to 10 (1)	pages 18- 18 6 lines
18	17	<i>Configuration.py</i>	sheets	10 to 10 (1)	pages 19- 19 9 lines
19	18	<i>checkFacturacion.py</i> .	sheets	11 to 11 (1)	pages 20- 20 58 lines
20	19	<i>check-facturacion-index.html</i>	sheets	11 to 12 (2)	pages 20- 22 1 lines
21	20	<i>app.yaml</i>	sheets	13 to 13 (1)	pages 23- 23 16 lines
22	21	<i>admin-query.yaml</i>	sheets	13 to 13 (1)	pages 24- 24 19 lines
23	22	<i>adminQuery.py</i>	sheets	14 to 14 (1)	pages 25- 26 130 lines
24	23	<i>admin-query-index.html</i>	sheets	14 to 15 (2)	pages 26- 28 1 lines